MICRO FOCUS

Silk Test 19.0

Silk Test Workbench Help

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

2018-06-06

Contents

Welcome to Silk Test Workbench 19.0

Welcome to Silk Test Workbench 19.0

- *Getting Started*
- *Documentation*
- *Product Suite*

What's New

- *What's New in Silk Test Workbench 19.0*
- *Release Notes*

Featured Sections

- *Creating Tests*
- *Playing Back Tests and Analyzing Results*
- *Debugging Tests*
- *Object Recognition*

Tutorials and Demonstrations

- *Visual Test Tutorial*
- *Script Tutorial*
- *Free Web-Based Training at our Training Store*

Code Samples

- *Language Reference*

Online Resources

- *Micro Focus Resource Page*
- *Micro Focus Home Page*
- *Micro Focus Channel on YouTube*
- *Silk Test Documentation*
- *Online Documentation*
- *Micro Focus SupportLine*
- *Micro Focus Product Updates*
- *Silk Test Knowledge Base*
- *Silk Test Forum*

Web-Based Training

Provide Feedback

Contacting Micro Focus on page 75
Email us feedback regarding this Help
Suggest a feature

Licensing Information

Unless you are using a trial version, Silk Test requires a license.

Note: A Silk Test license is bound to a specific version of Silk Test. For example, Silk Test 19.0 requires a Silk Test 19.0 license.

The licensing model is based on the client that you are using and the applications that you want to be able to test. The available licensing modes support the following application types:

Licensing Mode	Application Type

Mobile Native	• Mobile web applications.
	  • Android
	  • iOS
	• Native mobile applications.
	  • Android
	  • iOS

Full	• Web applications, including the following:
	  • Apache Flex
	  • Java-Applets
	• Mobile web applications.
	  • Android
	  • iOS
	• Apache Flex
	• Java AWT/Swing, including Oracle Forms
	• Java SWT and Eclipse RCP
	• .NET, including Windows Forms and Windows Presentation Foundation (WPF)
	• Rumba
	• Windows API-Based
	Note: To upgrade your license to a Full license, visit *http://www.microfocus.com*.

Premium	All application types that are supported with a *Full* license, plus SAP applications. **Note:** To upgrade your license to a Premium license, visit *http://www.microfocus.com*.
Mobile Native Add-On	In addition to the technologies supported with a Full or Premium license, the mobile native add-on license offers support for testing native mobile applications on Android and iOS.

Getting Started

Silk Test Workbench is the ideal Silk Test client for business analysts, domain experts and automation experts who focus on ease-of use in their approach to testing. Silk Test Workbench provides a visual test option, offering a powerful and intuitive no-coding approach to creating tests visually. A scripting option provides additional power and flexibility and visual tests can be combined and enhanced with VB.NET scripts. With Silk Test Workbench, you can record user sessions with your applications to create tests, enhance the tests by adding verifications and test logic, and play back the tests to ensure that the applications work as expected.

This section provides an introduction to Silk Test Workbench and an overview of the Silk Test Workbench interface.

Logging On

1. Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Clients** > **Silk Test Workbench** or (in Microsoft Windows 10) **Start** > **Silk** > **Silk Test Workbench**. Silk Test Workbench starts and displays the **Login to SilkTest Workbench** dialog box.
2. From the **Authentication** list box, select if you want to use the **Silk Test Workbench** authentication, or the **Windows** authentication.

   Note: You can use the **Windows** authentication only for users that were added to the Silk Test Workbench users by using **Tools** > **Administration**.
3. If you are using the **Silk Test Workbench** authentication, type a user name in the **User Name** field and a password in the **Password** field. If you are using the **Windows** authentication, you do not have to provide a user name and password.

   Note: If you are using Silk Test Workbench for the first time, an administrative user name and password may be required. The default user name is `Admin` and the default password is `admin`. Change this password after logging on to prevent unauthorized access. You can change your logon password at any time.
4. Select the database to use from the **Database** list.

   For a database to appear in the **Database** list, you must first configure its database connection for use. To configure a database connection, click **Configure** on the **Logon** dialog box.
5. Click **OK**.

   If you are the first user that is logging in to this database, Silk Test Workbench will create a default project named *Project1* in the database. All assets that you create will be stored by default into this project. You can change the project name by clicking **Edit Project**.

   The **Start Screen** opens. If you are logging in to Silk Test Workbench for the first time, Silk Test Workbench selects *Project1* or the first available non-global object as the active object.

Changing Your Logon Password

Use the following procedure to change your logon password.

1. Click **Tools** > **Change Password**. The **Change Password** dialog box opens. The logon name of the current user displays in the title bar of the dialog box.
2. Change your password and then click **OK**. Changes take effect the next time the User ID is used to log on.

Introduction to Silk Test Workbench

Silk Test Workbench is an automated testing tool that accelerates the functional testing of complex applications developed with Microsoft, Java, Web, and many other distributed technologies. With Silk Test Workbench, you can record user sessions with your applications to create tests, enhance the test by adding verifications and test logic, and play back tests to ensure that the applications work as expected.

Silk Test Workbench offers two options for developing test automation. You can use:

- Visual tests
- VB-based .NET scripts

Silk Test Workbench reliably records user actions to quickly produce powerful functional visual tests. Each recorded visual test displays as a series of actions in clear, concise steps that can be easily understood by all testers, from novice to expert. Testers have broad control over how Silk Test Workbench records all user actions, such as processing orders or entering customer information. Testers can edit tests at any time during or after recording to meet the most difficult demands of any test project.

Silk Test Workbench provides support for testing applications developed in a wide variety of development tools. You can add test logic, real data, verifications, and error handling to any test for even greater flexibility and reliability.

Silk Test Workbench increases team productivity with project collaboration features to support efficient communication of test status between testing, development, Quality Assurance, and other project stakeholders. Ad-hoc query and detailed reporting allow teams to interpret test run results to help make informed test project decisions.

Benefits of Using Silk Test Workbench

Silk Test Workbench helps to plan, develop, and execute test procedures and identify faults on application software, Web applications, and mobile applications. With Silk Test Workbench, a user can record sessions with applications, add validation functions to ensure that actual results match anticipated results, and replay the sessions at any time to ensure an application works as expected.

Silk Test Workbench provides a range of benefits for development organizations:

Shorter development and testing cycles	Because Silk Test Workbench can playback tests unattended at night or on weekends when computer time is easier to schedule, it can accelerate development and testing schedules, helping applications reach the market faster.
Consistency	Because automated tests run the same way each time, Silk Test produces consistent, reproducible results.
Increased productivity	Because Silk Test Workbench can playback tests without manual intervention, programmers and testing staff are freed for other tasks.
Rapid test development	Create useful tests quickly and easily by recording user actions while navigating through and testing an application. Set up templates to provide a standard framework for tests. Automated tests run in a fraction of the time that it takes to manually test.
Understand test results	The details of test script and visual test playback are recorded in a result, which can be reviewed at any time.

Automation Under Special Conditions (Missing Peripherals)

Basic product orientation

Silk Test Workbench is a GUI testing product that tries to act like a human user in order to achieve meaningful test results under automation conditions. A test performed by Silk Test Workbench should be as valuable as a test performed by a human user while executing much faster. This means that Silk Test Workbench requires a testing environment that is as similar as possible to the testing environment that a human user would require in order to perform the same test.

Physical peripherals

Manually testing the UI of a real application requires physical input and output devices like a keyboard, a mouse, and a display. Silk Test Workbench does not necessarily require physical input devices during test replay. What Silk Test Workbench requires is the ability of the operating system to perform keystrokes and mouse clicks. The Silk Test Workbench replay usually works as expected without any input devices connected. However, some device drivers might block the Silk Test Workbench replay mechanisms if the physical input device is not available.

The same applies to physical output devices. A physical display does not necessarily need to be connected, but a working video device driver must be installed and the operating system must be in a condition to render things to the screen. For example, rendering is not possible in screen saver mode or if a session is locked. If rendering is not possible, low-level replay will not work and high-level replay might also not work as expected, depend on the technology that is used in the application under test (AUT).

Virtual machines

Silk Test Workbench does not directly support virtualization vendors, but can operate with any type of virtualization solution as long as the virtual guest machine behaves like a physical machine. Standard peripherals are usually provided as virtual devices, regardless of which physical devices are used with the machine that runs the virtual machine.

Cloud instances

From an automation point of view, a cloud instance is not different to a virtual machine. However, a cloud instance might run some special video rendering optimization, which might lead to situations where screen rendering is temporarily turned off to save hardware resources. This might happen when the cloud instance detects that no client is actively viewing the display. In such a case, you could open a VNC window as a workaround.

Special cases

Application launched without any window (headless)	Such an application cannot be tested with Silk Test Workbench. Silk Test Workbench needs to hook to a target application process in order to interact with it. Hooking is not possible for processes that do not have a visible window. In such a case you can only run system commands.
Remote desktops, terminal services, and remote	If Silk Test Workbench resides and operates within a remote desktop session, it will fully operate as expected.  **Note:** You require a full user session and the remote viewing window needs to be maximized. If the remote viewing window is not displayed for some reason, for example network issues, Silk Test Workbench will continue to replay but might produce unexpected results, depending on what remote viewing

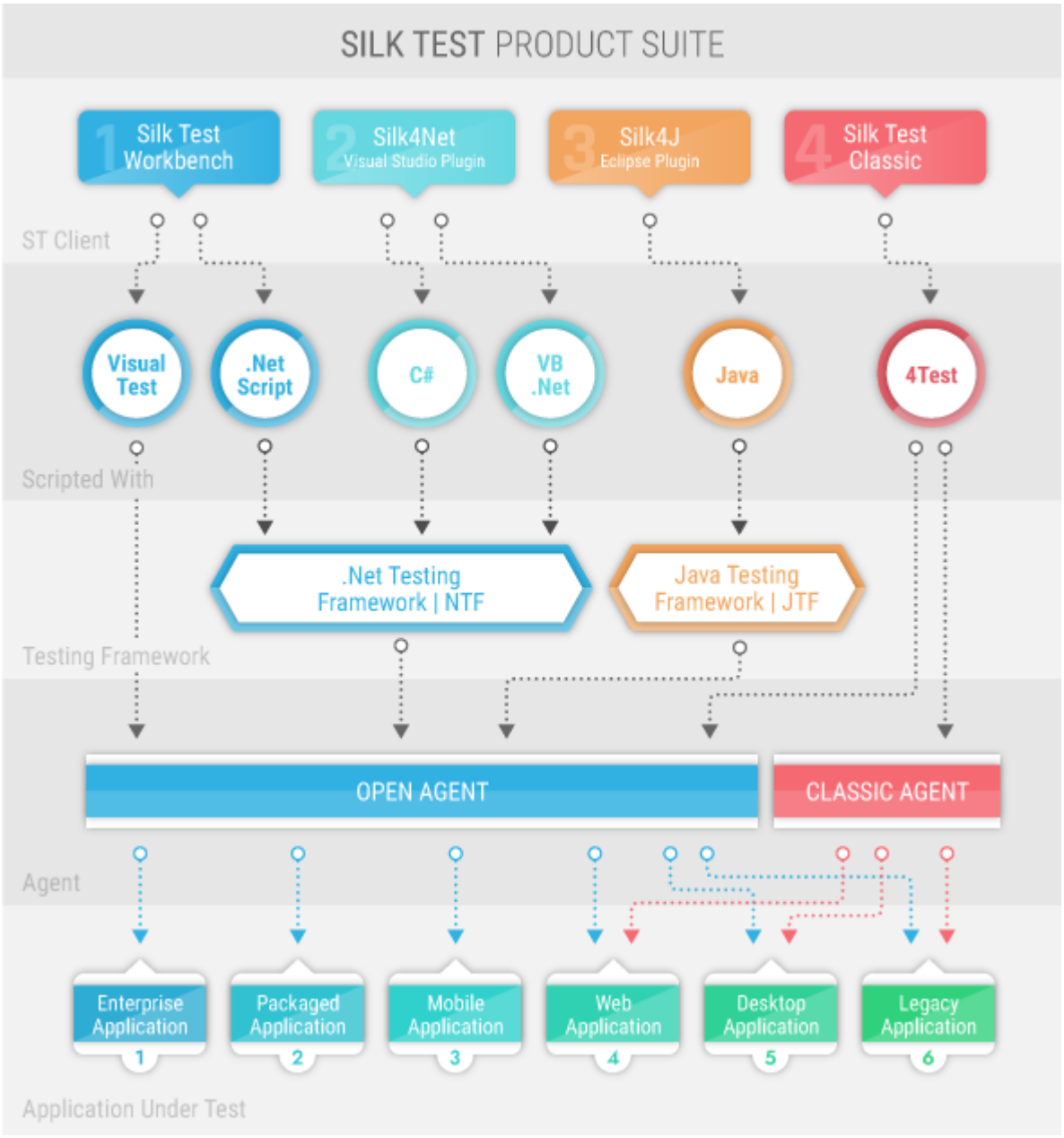
applications (all vendors)	technology is used. For example, a lost remote desktop session will negatively impact video rendering, whereas other remote viewing solutions might show no impact at all once the viewing window was lost.
	If Silk Test Workbench is used to interact with the remote desktop, remote view, or remote app window, only low-level techniques can be used, because Silk Test Workbench sees only a screenshot of the remote machine. For some remote viewing solutions even low-level operations may not be possible because of security restrictions. For example, it might not be possible to send keystrokes to a remote application window.
Known automation obstacles	Silk Test Workbench requires an interactively-logged-on full-user session. Disable anything that could lock the session, for example screen savers, hibernation, or sleep mode. If this is not possible because of organizational policies you could workaround such issues by adding *keep alive* actions, for example moving the mouse, in regular intervals or at the end of each test case.
	Note: Depending on the configuration of the actual testing environment and the technologies that are used for the AUT, the virtualization, and the terminal services, you may face additional challenges and limitations during the test automation process.

Silk Test Product Suite

Silk Test is an automated testing tool for fast and reliable functional and regression testing. Silk Test helps development teams, quality teams, and business analysts to deliver software faster, and with high quality. With Silk Test you can record and replay tests across multiple platforms and devices to ensure that your applications work exactly as intended.

The Silk Test product suite includes the following components:

- Silk Test Workbench – Silk Test Workbench is the quality testing environment that offers .NET scripting for power users and easy to use visual tests to make testing more accessible to a broader audience.
- Silk4NET – The Silk4NET Visual Studio plug-in enables you to create Visual Basic or C# test scripts directly in Visual Studio.
- Silk4J – The Silk4J Eclipse plug-in enables you to create Java-based test scripts directly in your Eclipse environment.
- Silk Test Classic – Silk Test Classic is the Silk Test client that enables you to create scripts based on 4Test.
- Silk Test Agents – The Silk Test agent is the software process that translates the commands in your tests into GUI-specific commands. In other words, the agent drives and monitors the application you are testing. One agent can run locally on the host machine. In a networked environment, any number of agents can run on remote machines.



The sizes of the individual boxes in the image above differ for visualization purposes and do not reflect the included functionality.

The product suite that you install determines which components are available. To install all components, choose the complete install option. To install all components with the exception of Silk Test Classic, choose the standard install option.

How Silk Test Workbench Works

Silk Test Workbench mimics the actions of a human tester. It manipulates an application the same way a tester would by sending keystrokes and mouse actions and by selecting items from menus and lists. However, keystrokes and mouse actions are automatically generated by the software, rather than by hardware. Silk Test Workbench can determine test actions by monitoring the display to see how an application responds to inputs.

You can flag test conditions where the expected result and actual result do not match. Silk Test Workbench tests are flexible. They can determine which action to perform based on the result of the previous action. Silk Test Workbench also has access to the PC's internal resources, such as the clock and the hard disk giving it a sense of time as well as the ability to read and write files.

The **Start Screen** and **Visual Navigator** are designed to help you start providing test solutions quickly. The **Start Screen** enables you to manage your test projects while the **Visual Navigator** offers control of individual test steps.

The centralized multi-user asset database helps manage the testing process. It also provides users with reporting capabilities for test executions.

Silk Test Workbench uses dynamic object recognition to interact with a test application. Dynamic object recognition enables you to create tests that use Silk Test Workbench queries to find and identify objects. Silk Test Workbench assigns each dialog box and control a locator name that uniquely identifies it. Silk Test Workbench then uses the locator name to locate the windows, dialog boxes, and controls in the application. Similarly, a Web page may have many controls, such as buttons, to which Silk Test Workbench also assigns a unique locator name.

With Silk Test Workbench, you can test:

- An application that runs on a different machine than Silk Test.
- Applications running on two or more machines at once.
- Multiple different applications simultaneously.
- Database operations of the test application.
- Mobile applications.

Silk Test Workbench UI

Silk Test Workbench's desktop is the starting point for all test activities. With the exception of the menu bar, you can control the display and appearance of the windows, panes, and other features on the desktop.

The main parts of the Silk Test Workbench UI are the following:

Menu Bar	Contains all menus that are available for Silk Test Workbench. For additional information about the available menus and menu commands, see *Menus*.
Toolbars	The toolbars provide one-click access to commonly used actions. For additional information about the available actions in the toolbars, see *Toolbars*.
Start Screen	The **Start Screen** is your launching point into the functionality of Silk Test Workbench, enabling you to quickly begin creating test solutions for all your applications. For additional information, see *Start Screen*.
Asset Browser	Use the **Asset Browser** to manage test assets. The **Asset Browser** provides a single point for creating, managing, and viewing assets for each asset type in the database. For additional information, see *Asset Browser*.
Visual Navigator	The **Visual Navigator** graphically represents the elements of a visual test and allows you to interact with each element through a point-and-click interface. The **Visual Navigator** is only visible when you create or open a visual test. For additional information, see *Visual Navigator*.
Code Window	Use the **Code** window to record, design, and modify scripts. The **Code Window** is only visible when you create or open a VB .NET script. For additional information, see *Code Window*.

Silk Test Workbench Software Components

Silk Test Workbench consists of two distinct software components that execute in separate processes:

- The *Silk Test Workbench host software* is the component you use to develop, edit, compile, run and debug your scripts and visual tests. The machine that runs this program is often referred to as the host machine.
- The *Silk Test agent* is the component of Silk Test Workbench that interacts with the GUI of your application. The agent translates the commands in your tests into GUI specific commands, driving and monitoring the application you are testing. The agent can be run locally on the same machine on which the host is running or, in a networked environment, any number of agents can be run on remote machines. The machine that runs the agent is often referred to as the remote machine.

If You Are New to Automated Testing

Software testing is an integral part of the software development process. Before software can be released, testers must verify that it performs as expected, often on different types of computers or different operating systems. Consequently, software testing has become increasingly complex, requiring repetitive testing of features and functions on different computers. But as the complexity for testing software has increased, the time allocated to testing software has typically been reduced to meet release schedules. Automated testing allows you to test software for bugs and performance problems effectively and efficiently, making it an essential tool for production of quality software.

To test an application manually, you create a series of test cases that specify how the test application should react when you perform certain actions, such as pressing sequences of keys, choosing menu options, or using your mouse. A manual test cycle typically follows a pattern similar to the following:

- Take your test application to the starting point of the test.
- Perform a series of actions that interact with the test application.
- Observe what happens as you interact with the application.
- Compare the results of your interaction with what you expected would happen.
- Record any discrepancies between what you expected and what you saw.
- Fix the application as needed.
- Retest the application or move on to the next test.

Silk Test Workbench can mimic each step of this test pattern; it can type on your keyboard, move the mouse and click mouse buttons, compare what is displayed on the computer to what is expected, and save results of tests to a file for subsequent analysis. You can link tests, so that the successful conclusion of one test triggers the start of the next test. Because actions and verifications are included, automated testing is faster and more reliable than manual testing.

Testing Strategy

Before creating visual tests, scripts, and other Silk Test Workbench assets to build application testing solutions, it is a good practice to plan a testing strategy.

It is not necessary to include all the parts of a specific test solution in a single visual test or script and is not usually beneficial to do so.

Typically, the most efficient testing approach is a modular approach. Think of your application testing in terms of distinct series of transaction units.

For example, testing an online ordering system might include the following distinct transaction units:

- Log on to the online system
- Create a customer profile
- Place orders
- Log off the online system

If one test is created to handle all of these distinct units and there are ten different scenarios that use this test, you would need to record ten different tests to handle the scenarios. If any change occurs to the application, for example if an extra field is added to the logon window, ten different tests would require a change to accommodate data input to the new field.

Rather than creating one visual test or script that tests all of these transaction units, and then recreating it ten times for each scenario, it may be more beneficial to create separate tests as test "modules" that handle each one of these transaction units. If a separate test is created for each of the separate transaction units and reused for each of the test scenarios, then only the test that handles the logon transaction unit would require change.

When you run the driver script, it calls and runs each script in succession.

With a modular approach, you can create one additional test to test stock validation, then update the driver script to call it in its appropriate sequence. The following example shows a driver script that runs separate scripts to test the online ordering system.

```
Workbench.RunScript ("OnlineLogOn_Script")
Workbench.RunScript ("CreateProfile_Script")
Workbench.RunScript ("ValidateStock_Script")
Workbench.RunScript ("PlaceOrders_Script")
Workbench.RunScript ("OnlineLogOff_Script")
```

Implementing a modular testing strategy in this manner also makes it easier to facilitate changes in the test flow.

For example, if an additional transaction unit is required to test the online ordering system such as validating stock prior to placing orders, you need to change ten tests to handle stock validation for the ten different test scenarios.

This modular approach enhances reusability but does not necessarily accommodate different test "scenarios", where a testing strategy involves repeating a sequence of transaction units multiple times, using different data each time a transaction is run. ActiveData enables you to use external data sources to enhance the reusability of modular test scripts in application testing.

Functions can also be called from driver scripts to test solutions that can be reused in different application tests.

Using a Database in a Remote Location

When using Silk Test Workbench in a distributed environment, which means that a central Silk Test Workbench data base is accessed by the Silk Test Workbench clients over a wide area network, you might face one or more of the following issues:

- High latency between the Silk Test Workbench clients and the data base server.
- Increased network traffic because of many requests to the data base.

In such a distributed environment, Micro Focus recommends applying the following best practice:

- Keep project sizes small, to prevent huge amounts of data being transferred between the Silk Test Workbench clients and the data base server, for example when using the **Asset Browser**.
- Avoid viewing test results remotely.
- Whenever possible, work offline and export projects from the central data base. For this, Silk Test Workbench users would have to coordinate regarding which assets should be treated as read-only and what assets are updated by whom. Additionally, any assets from the central data base would have to be imported to the remote clients.
- Use virtual machines and remote desktop connections to access the central data base.

Silk Test Workbench Documentation

When Silk Test Workbench is installed on your machine, you can access the documentation by clicking (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Documentation** or (in Microsoft Windows 10) **Start** > **Silk**.

The Silk Test Workbench documentation set includes the following:

Silk Test Installation Guide	Provides system requirements and instructions for installing Silk Test Workbench. It also details basic setup and how to create, configure, and connect to a Silk Test Workbench database. You can also access the *Silk Test Installation Guide* online from the *Silk Test Documentation*.
Silk Test Release Notes	For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.
Silk Test Workbench Help	Provides Silk Test Workbench product usage information and the Silk Test Workbench language reference.
Silk Test Tutorials	These tutorials provide information on testing specific application types with Silk Test and on using specific features of Silk Test. In addition to the tutorials located in your local installation folder, you can also download the latest tutorials from the *Silk Test Documentation*.

What's New in Silk Test Workbench

Silk Test Workbench supports the following new features:

Shared Projects in Silk Test Workbench

Silk Test Workbench 19.0 introduces project dependencies to assist you in saving time and effort when handling complex testing requirements.

With this new functionality, Silk Test Workbench provides enhanced support for testing enterprise applications across multiple versions. The new project dependencies also provides enhanced reuse capabilities of existing projects and enable you to easily test multiple hotfixes or other minor versions of your applications.

Cross-Browser Testing on a Mac

You can now record and replay tests for web applications against Mozilla Firefox or Google Chrome on macOS.

Customization of Recording

Creating good identifiers for the items in your application under test is challenging. While Silk Test Workbench uses sophisticated strategies to automatically generate the best possible locators and object map entry names during recording, these might not always cover your specific requirements. For example, the generated locator might not include an attribute that you want to be included, and the automatically generated object map entry for a recorded object might be difficult to read or might contain special characters. With Silk Test Workbench 19.0, you can edit the locator and the object map entry for web controls directly from the recording UI.

This new functionality is available for all supported desktop and mobile browsers except Internet Explorer.

Usability Enhancements

This section lists usability enhancements that have been made in Silk Test 19.0.

Faster test execution with parallel testing

To enable you to execute more tests faster, parallel testing is now enabled by default for all Silk Test clients.

Capturing Screenshots of Entire Web Pages

In addition to capturing screenshots of the currently visible portion of a web page during testing, Silk Test now offers the ability to take a screenshot of all contents of a web page by using the new `CaptureFullPageBitmap` method of the `BrowserWindow` class.

Stopping the Open Agent from the command line

You can now use the -shutDown option to stop the Open Agent from the command line.

Disabling iframe and frame support for browser testing

To improve the replay performance of your browser tests, you can now disable the iframe and frame support whenever you are not interested in the content of the iframes and frames in a web application. For example, disabling the iframe support might significantly improve replay performance for web pages with many ads and when testing in a mobile browser. You can disable iframe and frame support for all browsers except Internet Explorer.

Enhanced Support for Testing with Multiple UI Sessions

Instead of using `MicroFocus.SilkTest.MultiSessionLauncher.exe`, you can now directly start the Open Agent from a UI session with the following call:

```
openAgent.exe -infoServicePort=<port>
```

Getting or setting additional options from a visual test

You can now get or set the values of the options under **Tools** > **Options** > **Advanced** from a visual test.

Enhanced Silk Test Workbench database maintenance

While maintenance tasks must be performed on Silk Test Workbench, the administrator can now turn on the maintenance mode and log out non-administrator users from the Silk Test Workbench database. Additionally, all Silk Test Workbench users can now use the **Connected Users** window to see which users are currently connected to the database.

Capturing screenshots for failing verifications

Silk Test Workbench now supports capturing a screenshot in the result file whenever a verification in a visual test or a .NET script fails.

Accessing the definition of a class or method

You can now select **Go to Definition** in the context menu of a VB .NET script in Silk Test Workbench to open the definition of the selected class or method.

Specifying the project into which to import assets

For each source project included in an exported assets ZIP file, you can now specify during import into which target project the assets in the source project should be imported.

API Enhancements

This section lists API enhancements that have been made in Silk Test 19.0.

Support for additional XPath axes

By supporting the XPath axes ancestor, preceding-sibling, and following-sibling, Silk Test now allows locating controls based on the properties of a descendant control or by the properties of a sibling.

Check whether a DOM object is focused

You can now use the `IsFocused` property to check whether the specified DOM object is in focus or not.

Mobile Testing Enhancements

Silk Test Workbench now supports testing toasts for native mobile applications on Android.

Technology Updates

This section lists the significant technology updates for Silk Test 19.0.

New Mozilla Firefox Versions

In addition to the versions of Mozilla Firefox, which have been tested with the previous version of Silk Test, Silk Test has now been tested with the following new versions of Mozilla Firefox:

- Mozilla Firefox 57
- Mozilla Firefox 58
- Mozilla Firefox 59
- Mozilla Firefox 60

Note: This list includes the new versions of Mozilla Firefox that have been tested with Silk Test 19.0 until the release date of Silk Test 19.0. Silk Test 19.0 should be able to support newer versions of Mozilla Firefox, even if these versions have been released after the release date of Silk Test 19.0.

New Google Chrome Versions

In addition to the versions of Google Chrome, which have been tested with the previous version of Silk Test, Silk Test has now been tested with the following versions of Google Chrome:

- Google Chrome 63
- Google Chrome 64
- Google Chrome 65
- Google Chrome 66
- Google Chrome 67

Note: This list includes the versions of Google Chrome that have been tested with Silk Test 19.0 until the release date of Silk Test 19.0. Silk Test 19.0 should be able to test with newer versions of Google Chrome, even if these versions have been released after the release date of Silk Test 19.0.

New Microsoft Edge Version

In addition to the versions of Microsoft Edge, which have been tested with the previous version of Silk Test, Silk Test has been tested for recording and replay with Microsoft Edge 42.17134, the Microsoft Edge version for the Windows 10 April 2018 Update.

Java 9 and 10 Support

Silk Test Workbench now supports testing applications that are based on Java 9 and Java 10.

Enhanced MFC Support

Silk Test Workbench now generates stable locators for Microsoft Foundation Class (MFC) controls.

Managing Projects

A project is a collection of assets with assigned access privileges for each user on each project. One of your tasks as an administrator of the Silk Test Workbench environment is to maintain the list of projects.

Use projects to store assets in a logical association. For example, all assets associated with a specific application or build can be stored together in a project.

Administrators can create and add projects to the Silk Test Workbench database. When creating a project, the access rights of the creator to the new project reflect the default access rights that are granted to the creator. Storing assets in separate projects provides different levels of security. For example, a user might have full access for *Project1*, but read-only access for *Project2*, and no access for *Project3*.

Only the assets in the projects for which the current user has access to can be viewed. Assets are viewed and managed using the **Asset Browser**.

Note: To optimally use the functionality that Silk Test Workbench provides, create an individual project for each application that you want to test, except when testing multiple applications in the same test.

Common Project

The default project is the Common project, which serves as a central repository for assets used in multiple projects. Assets in the Common project are available to all other projects. The Common project is the active project by default.

Active Project

Newly created assets are added to the project that is set as the Active project. Assets added to the Active project are available only in that project and in all projects that reference the Active project.

Note: Assets in a project can only access other assets in the same project or in a referenced project.

Viewing Projects

The **Asset Browser** displays all test assets that belong to the active project and to the projects that are selected in the **View** dialog box. You can only view the test assets in projects for which you have access rights.

By default, new assets are always added to the active project and are available only in that project.

1. Choose **View** > **Asset Browser** or navigate to **Get Started** > **Tasks** on the **Start Screen** and click **Asset Browser**. The **Asset Browser** opens.
2. To change the active project, click **View** in the **Asset Browser** toolbar. The **View Assets from Projects** dialog box opens and displays the list of projects to which the current user has access.
3. Check the check boxes that correspond to the projects the assets of whom you want to view and click **OK**. The view of the **Asset Browser** changes to reflect the test assets contained in all the selected projects. The assets in the active project are always displayed.

Shortcut Keys

In addition to standard Windows menu commands, many commands have a key combination associated with it that lets you execute the command with keystrokes. Typically, you press the `Ctrl` or `Alt` key and simultaneously press the underlined letter from the appropriate command menu.

Menu Shortcut Keys for Commands

Press	To
Alt+F9	Using the selected object, insert a verification into a visual test or script during recording.
Alt+F10	Start/stop recording.
Alt+Enter	Display the **Summary Information** dialog box from within a visual test or script.

Press	To
Ctrl+Alt	Insert a verification into a visual test, during recording or editing, or into a script during recording.
Ctrl+F8	Plays back a visual test to the currently selected test step or a script to the currently selected statement.
Ctrl+F9	Executes playback to the next statement or pointer. Enabled when in debug mode.
Ctrl+K	Disable one or more currently selected lines in a script by commenting them out. You can use this shortcut to comment out entire blocks of code. You can also use **Ctrl+F11**.
Ctrl+N	Displays the **New** window, where you can create any type of asset.
Ctrl+O	Opens the **Asset Browser** to allow you to open an asset. Highlights the asset type last worked with.
Ctrl+P	Displays the print dialog box, where you can print the selected asset.
Ctrl+S	• Saves the current state of the asset as a new version of the asset and assigns an incremental numerical identifier to the version. The shortcut key combination is enabled if the default save behavior is set to **Save as new version**. For more information, see *Setting the Default Behavior for Saving Assets*. • Saves the current state of an asset as the current version of the asset. The shortcut key combination is enabled if the default save behavior is set to **Save as current version**. For more information, see *Setting the Default Behavior for Saving Assets*. This command is disabled the first time you save an asset.
Ctrl+Y	Reverses the previous **Undo** action in a test script. Not available for visual tests.
Ctrl+Z	Reverses the previous edit in a test script. Not available for visual tests.
Ctrl+Alt+A	Opens the **Asset Browser**.
Ctrl+Alt+I	Opens the **Identify Object** dialog box where you can record a locator for a selected object.
Ctrl+Alt+S	Displays the **Start Screen**.
Ctrl+Shift+F2	View last position.
Ctrl+Shift+F8	Executes all remaining steps in a visual test being played back from another visual test, then suspends playback at the next step in the original visual test. For scripts, executes all remaining code in a procedure as if it were a single statement, and exits to the next statement in the procedure that caused the procedure to be initially called.
Ctrl+Shift+F9	Clears all breakpoints in all open visual tests and VB .NET scripts.
Ctrl+Shift+K	Re-enable one or more selected lines in a script by un-commenting them.
Ctrl+Shift+L	• Saves the current states of all open assets as new versions of the assets and assigns an incremental numerical identifier to the version for each asset. The shortcut key combination is enabled if the default save behavior is set to **Save as new version**. For more information, see *Setting the Default Behavior for Saving Assets*. • Saves the current states of all open assets as the current version of the assets. The shortcut key combination is enabled if the default save behavior is set to **Save as current version**. For more information, see *Setting the Default Behavior for Saving Assets*. This command is disabled the first time you save an asset.
Ctrl+Tab	Switch between open windows in the user interface.
Ctrl+Space	Complete word.

Press	To
Del	Deletes the selected item.
F1	Display help for a specific item.
F2	Display the **Edit Script Input Parameter** or the **Edit Script Output Parameter** dialog box, depending on what type of parameter is selected in the **Properties** pane.
F4	View properties.
F5	Playback the active visual test or script.
F7	Checks that the script syntax meets all preconditions for compiling, compiles the code, and shows any errors that occur.
F8	Plays back a visual step one test step at a time or a script one statement at a time.
F9	Sets a breakpoint at the selected step or line of code or clears an existing breakpoint from the selected step or line of code.
Shift+F2	View definition.
Shift+F8	Plays back an embedded visual test or script in its entirety and suspends in debug mode at the next step in the original visual test or script. Also steps over a function call when pressed while in a VB .NET script.

Tutorials

The Silk Test Workbench tutorials are self-paced guides for learning the basics of using Silk Test Workbench to test your applications. In addition, the tutorials introduce several advanced features that enable you to create powerful and flexible tests that handle varying conditions in your test application and report back the details of exactly what happened during playback.

The tutorials provide a hands-on demonstration in which you interact directly with Silk Test Workbench to test a sample application using a real world scenario.

Note: The sample applications used in the Silk Test Workbench tutorials are designed and optimized to run on Internet Explorer. To ensure a user experience consistent with the lessons in the tutorials, we do not recommend running the tutorial sample applications on another one of the supported browsers instead of Internet Explorer.

Silk Test Workbench Visual Test Tutorial

Welcome to the Silk Test Workbench Visual Test tutorial, a self-paced guide that demonstrates how to use Silk Test Workbench's visual, storyboard-based interface to create powerful and flexible functional tests. In this tutorial, you will learn the basic steps required to create a visual test, play back the visual test, and then analyze the results of the playback. Additionally, you will learn how to use a number of features that allow you to quickly update and enhance a recorded visual test.

This tutorial uses the Silk Test sample Web application, *http://demo.borland.com/InsuranceWebExtJS/*, to create a real world scenario in which you practice using Silk Test Workbench to create repeatable tests.

The lessons in this tutorial are designed to be completed in sequence as each lesson is based on the output of previous lessons.

Introducing the GUI

This section introduces the GUI including the Main Screen, Start Screen, and Visual Navigator. This section is optional. If you are already familiar with the basic elements of the development environment, proceed to the next section.

Silk Test Workbench Main Screen

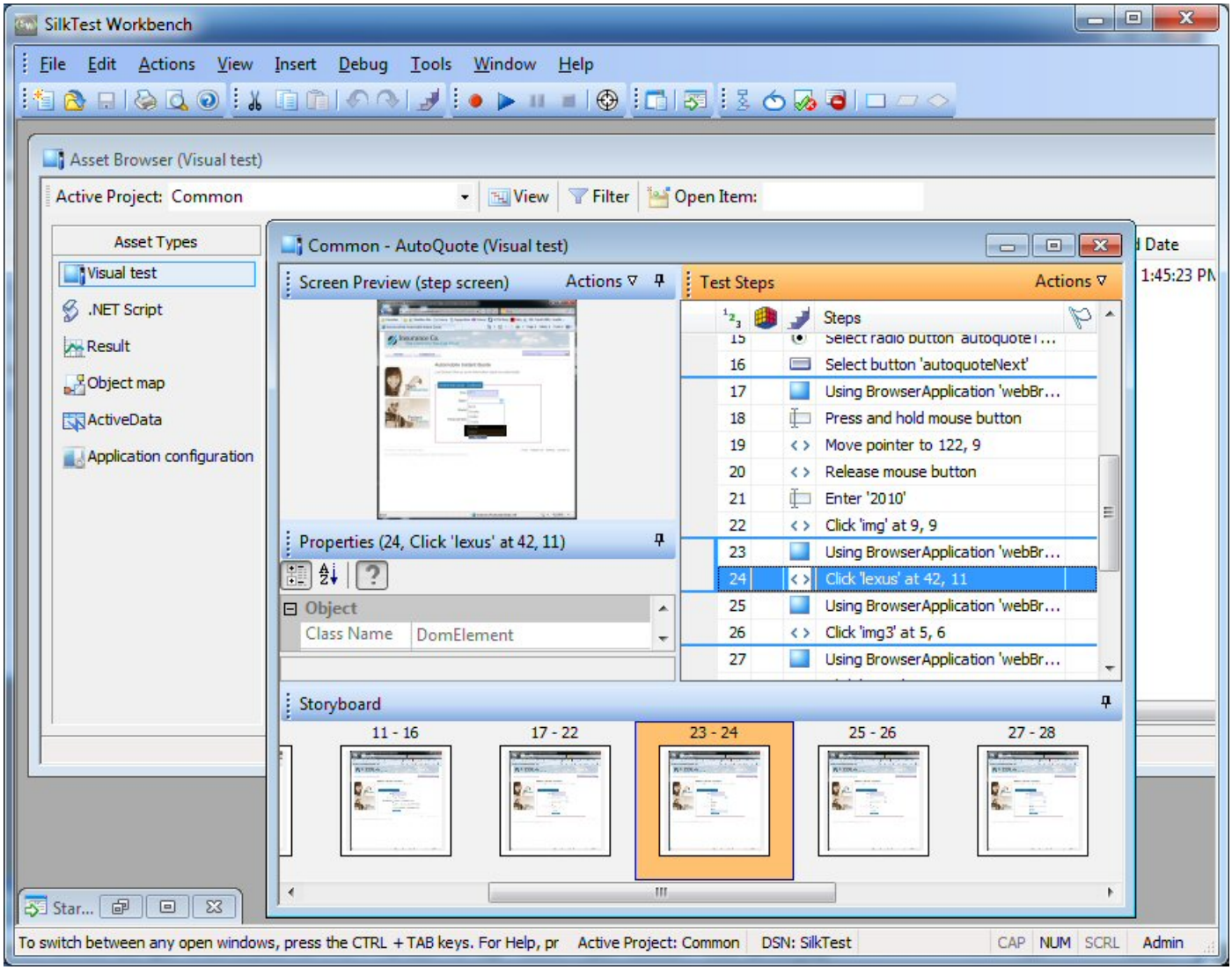
Silk Test Workbench's main screen is the starting point for all test activities. With the exception of the menu bar, the user maintains control over all of the items that are displayed on the desktop.

All other Silk Test Workbench windows, including the Start Screen, Visual Navigator, and Asset Browser, display as child windows in the Silk Test Workbench Main Screen.

The Silk Test Workbench Main Screen contains the following:

- Menu bar
- Toolbars
- Main asset viewing area
- Status bar

The following graphic shows these elements in the Silk Test Workbench Main Screen. The main asset viewing area shows a visual test in the Visual Navigator, the Asset Browser in the background, and the Start Screen, which is minimized.



Start Screen

The **Start Screen** is your launching point into the functionality of Silk Test Workbench, enabling you to quickly begin creating test solutions for all your applications.

To enable you to quickly start testing your applications with minimal effort, the **Start Screen** includes the following areas:

Area	Description
Start	Enables you to record or open visual tests, .NET scripts, and keyword-driven tests with a single click.
Assets	Enables you to open recently used visual tests, .NET scripts, and keyword-driven tests with a single click. The recent assets list is updated in real time so any changes or updates will move the current asset to the top of the list. You can play back an asset or view eventual results of previous executions of the asset by clicking on the corresponding icons to the right of the asset.  **Note:** The **Start Screen** settings under **Tools** > **Options** > **Start Screen** allow you to control how many items display in the recent assets list.
What's New	Lists new features and enhancements that are included in the current version of Silk Test Workbench. Also lists available updates for Silk Test Workbench.  **Note:** If you have opted not to display the start screen when you start Silk Test Workbench, you can check for available updates by clicking **Help** > **Check for Product Update**.
Get Started	Provides quick links to commonly used tasks and useful help topics to get you quickly productive.
Resources	Provides links to information that is related to Silk Test Workbench, for example to the support knowledge base, the Silk Test community, and so on.
News & Community	Provides links to the latest forum entries and blog posts on the Silk Test community.
Flags	Displays the test steps in visual tests and results that are flagged and assigned for follow-up. Use the **Flags** pane to collaborate with other testers and other users of the test projects in the database, by exchanging information about test suites and test projects. By default, flags are grouped by the date they were created and the asset they appear in. To quickly view information about a flag, move your pointer over a flag. A ToolTip displays containing the flag description, modification data, the project, and the ID to whom the flag is assigned. You can change how flags are displayed by modifying the **Start Screen Flag** options under **Tools** > **Options** > **Start Screen** > **Flags**. Double-click a flag in the **Flags** pane to open its associated visual test or result.  **Note:** The **Flags** pane is only visible in the **Start Screen** if at least one flag is assigned to you or if at least one flag is filtered by the settings to be shown for you.

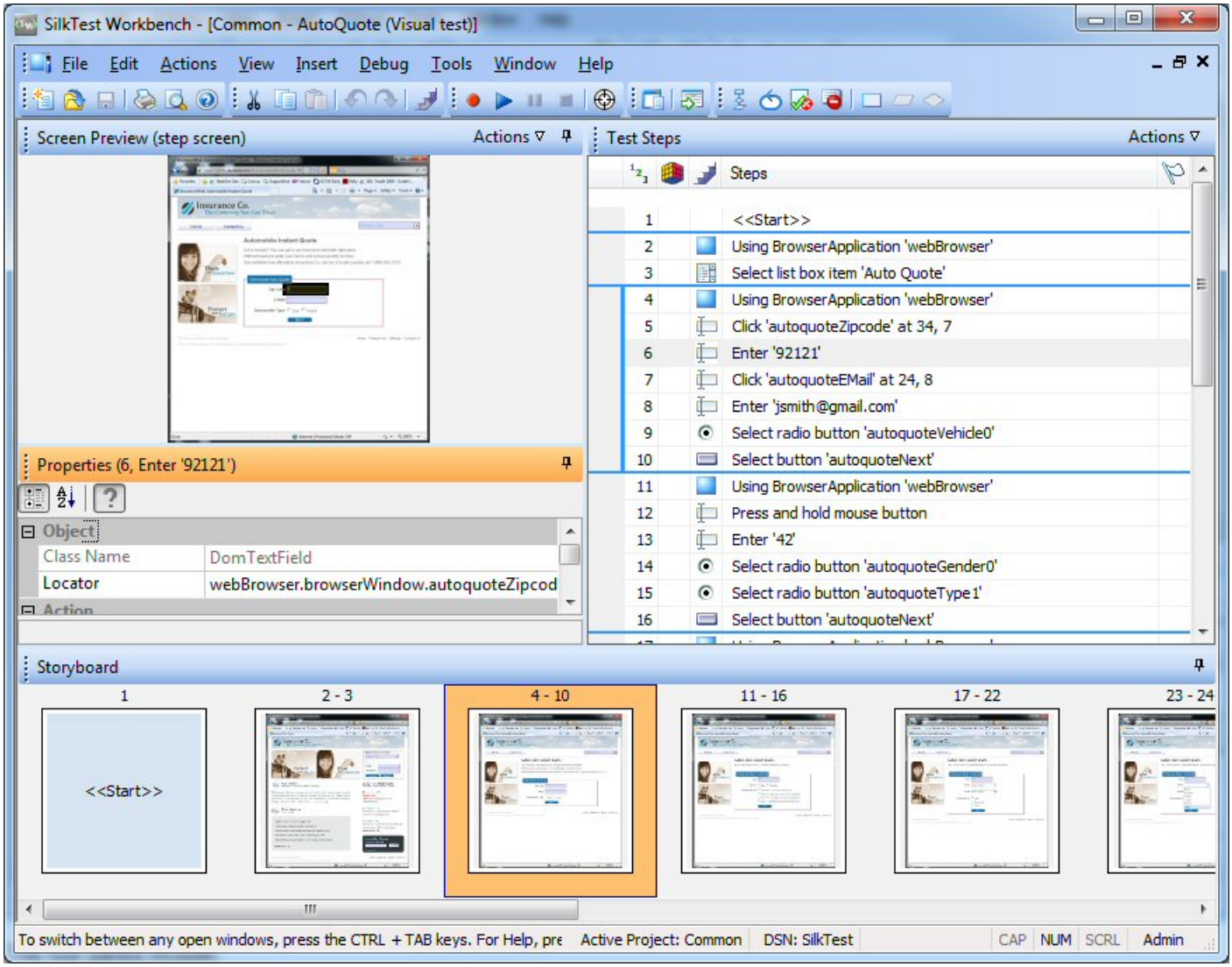
Click **Close Window** in the upper-right corner of the **Start Screen** to close the **Start Screen**.

 **Tip:** You can always access the **Start Screen** from the main screen by clicking **Start Screen** in the toolbar. You can also press `Ctrl+Alt+S`, or choose **View** > **Start Screen** at any time to display it.

 **Note:** The user interface works with standard Windows small and large font sizes. Using a custom font size may result in the inaccurate display of text in the user interface.

Visual Navigator

The **Visual Navigator** graphically represents the elements of a visual test and allows you to interact with each element through a point-and-click interface. When viewed in the **Visual Navigator**, a visual test is represented by information displayed in four panes that collectively provide a comprehensive view of each step in a visual test.



The four panes include:

Test Steps	Lists each step of a visual test in clear non-technical language.
Screen Preview	Displays a snapshot of the application under test as it appears when a step executes during playback of a visual test.
Properties	Displays the properties of a step in a visual test.
Storyboard	Displays the flow of a visual test through the use of thumbnail images, which represent the logical groups of steps in a visual test.

Note:

Silk Test Workbench takes snapshots under the following circumstances:

- Before every automation test step during recording.

  Note: For SAP applications, snapshots occur when the screen changes rather than before every automation test step.
- When executing a `Using` step in a visual test, a snapshot of the result.

- When a playback error occurs.

The **Screen Preview**, **Storyboard**, and **Properties** panes are synchronized with the **Test Steps** pane and display information specific to a selected step in the **Test Steps** pane. In this way, you can easily view all aspects of a step by selecting a step in the **Test Steps** pane, and then viewing information about the step in the other panes.

In addition to viewing a visual test, the **Visual Navigator** also allows you to enhance or update an existing visual test by using the **Screen Preview** and **Properties** panes. For example, in the **Properties** pane, after recording a visual test, you can change the literal value of a recorded property by replacing it with a variable. Additionally, to quickly update a visual test when changes occur in the application under test, you can update previously captured screens using the **Update Screen** feature of the **Screen Preview**.

The **Visual Navigator** also displays the playback result of a visual test using the same panes as those used for a visual test. For a result, the panes have additional functionality and appear in the **Result** window, which contains toolbar options and several tabs that display different views of result content. Examples of additional functionality specific to a result include the ability to see the pass or fail status of each step in the **Test Steps** pane. Additionally, in the **Screen Preview**, you can see a comparison of the differences between the screens captured during recording and screens captured during playback, and then update the existing visual test without accessing the test application.

Recording a Visual Test: Introduction

As you perform actions to create an insurance quote request in the sample Web application, Silk Test Workbench records the actions as steps. Steps form the basis of a visual test. When you have completed recording actions needed for a test, you can see the recorded test in the Visual Navigator. Steps display in the Test Steps pane of the Visual Navigator.

Note: The sample application used in this tutorial is designed and optimized to run on Internet Explorer. To ensure a user experience consistent with the lessons in the tutorial, Micro Focus does not recommend running the tutorial sample application on one of the other supported browsers instead of Internet Explorer.

Note: Before you record or playback web applications, disable all browser add-ons that are installed in your system. To disable add-ons in Internet Explorer, click **Tools** > **Internet Options**, click the **Programs** tab, click **Manage add-ons**, select an add-on and then click **Disable**.

Starting the Sample Web Application

For this tutorial, use the Silk Test sample Web application. This Web application is provided for demonstration purposes.

Use the Silk Test sample Web application with Internet Explorer. To ensure a user experience consistent with the lessons in the tutorial, we do not recommend running the sample Web application with one of the other supported browsers instead of Internet Explorer.

1. To record DOM functions to make your test faster and more reliable, perform the following steps:
   a) Click **Tools** > **Options**.
   b) Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
   c) Click **xBrowser**.
   d) From the **Record native user input** list box, select **No**.
   e) Click **OK**.

   Note: Typically, when you test Web applications, you use native user input rather than DOM functions. Native user input supports plug-ins, such as Flash and Java applets, and applications that use AJAX, while high-level API recordings do not.
2. Before you record or playback Web applications, you must disable all browser add-ons. To ensure that all browser add-ons are disabled, perform the following steps:

a) In Internet Explorer choose **Tools** > **Internet Options**. The **Internet Options** dialog box opens.
b) Click the **Programs** tab and then click **Manage add-ons**. The **Manage Add-ons** dialog box opens.
c) In the list of add-ons, review the **Status** column and ensure that the status for each add-on is **Disabled**.
   If the **Status** column shows **Enabled**, select the add-on and then click **Disable**.
d) Click **Close** and then click **OK**.

3. To access the sample application remotely, click *http://demo.borland.com/InsuranceWebExtJS*. The sample application Web page opens.

Recording a Visual Test for the Sample Web Application

During recording, Silk Test Workbench records all interactions in the test application (except interaction with Silk Test Workbench itself) until recording is stopped. After you have finished recording, you can modify the visual test you have generated to add and remove steps.

1. To be able to edit the recorded controls at a later point in this tutorial, enable the capturing of browser controls before recording the visual test.
   a) In the Silk Test Workbench menu, select **Tools** > **Options**. The **Options** dialog box opens.
   b) In the tree, select **Record** > **Output** > **Visual test**.
   c) Set the value of **Browser control capture** to **Yes**.

   
   Note: Enabling control capture during the recording of web applications might decrease the performance of your visual tests. Micro Focus recommends disabling control captures when you have completed the tasks described in this tutorial.
2. Choose **File** > **New**. The **New Asset** dialog box opens.
3. Select **Visual test** from the asset types list.
4. Check the **Begin Recording** check box to start recording immediately.
5. Click **OK** to save the visual test as an asset and begin recording. The **Select Application** dialog box opens.
6. Select the **Web** tab.
7. Select **Internet Explorer** from the list.
8. In the **Enter URL to navigate** text box, type `demo.borland.com/InsuranceWebExtJS/`.
9. Click **OK**. Silk Test Workbench minimizes and the **Recording** window opens.
10. In the Insurance Company Web site, perform the following steps:
    a) From the **Select a Service or login** list box, select **Auto Quote**. The **Automobile Instant Quote** page opens.
    b) Type a zip code and email address in the appropriate text boxes, click an automobile type, and then click **Next**.
       For example, type `92121` as the zip code, `jsmith@gmail.com` as the email address and specify `Car` as the automobile type.
    c) Specify an age, click a gender and driving record type, and then click **Next**.
       For example, type `42` as the age, specify the gender as `Male` and `Good` as the driving record type.
    d) Specify a year, make, and model, click the financial info type, and then click **Next**.
       For example, type `2010` as the year, specify `Lexus` and `RX400` as the make and model, and `Lease` as the financial info type.
       A summary of the information you specified appears.
    e) Click **Purchase**.
       The **Purchase A Quote** page opens.
    f) Click **Home** near the top of the page to return to the home page where recording started.
11. Stop recording by pressing `Alt+F10`, clicking **Stop** in the **Recording** window, or clicking the Silk Test Workbench taskbar icon. The **Recording Complete** dialog box opens. If the **Do not show this message again** check box is checked in the **Recording Complete** dialog box, this dialog box does not appear after recording is stopped. In this case, the visual test displays in the **Test Steps** pane.

Saving and Naming the Visual Test

Once you have recorded your test, the **Recording Complete** dialog box opens and then you can:

- Play back the recorded visual test.
- Review the recorded actions in the Visual Navigator.
- Save the visual test, and then review the recorded test in the Visual Navigator.

Since you have just recorded the visual test for the first time, save and name the test before playing it back or reviewing the recorded actions.

1. From the **Recording Complete** dialog box, click **Save**.

   Tip: When you create an asset without naming it, Silk Test Workbench assigns the temporary name *"Untitled_"* followed by a sequential number.

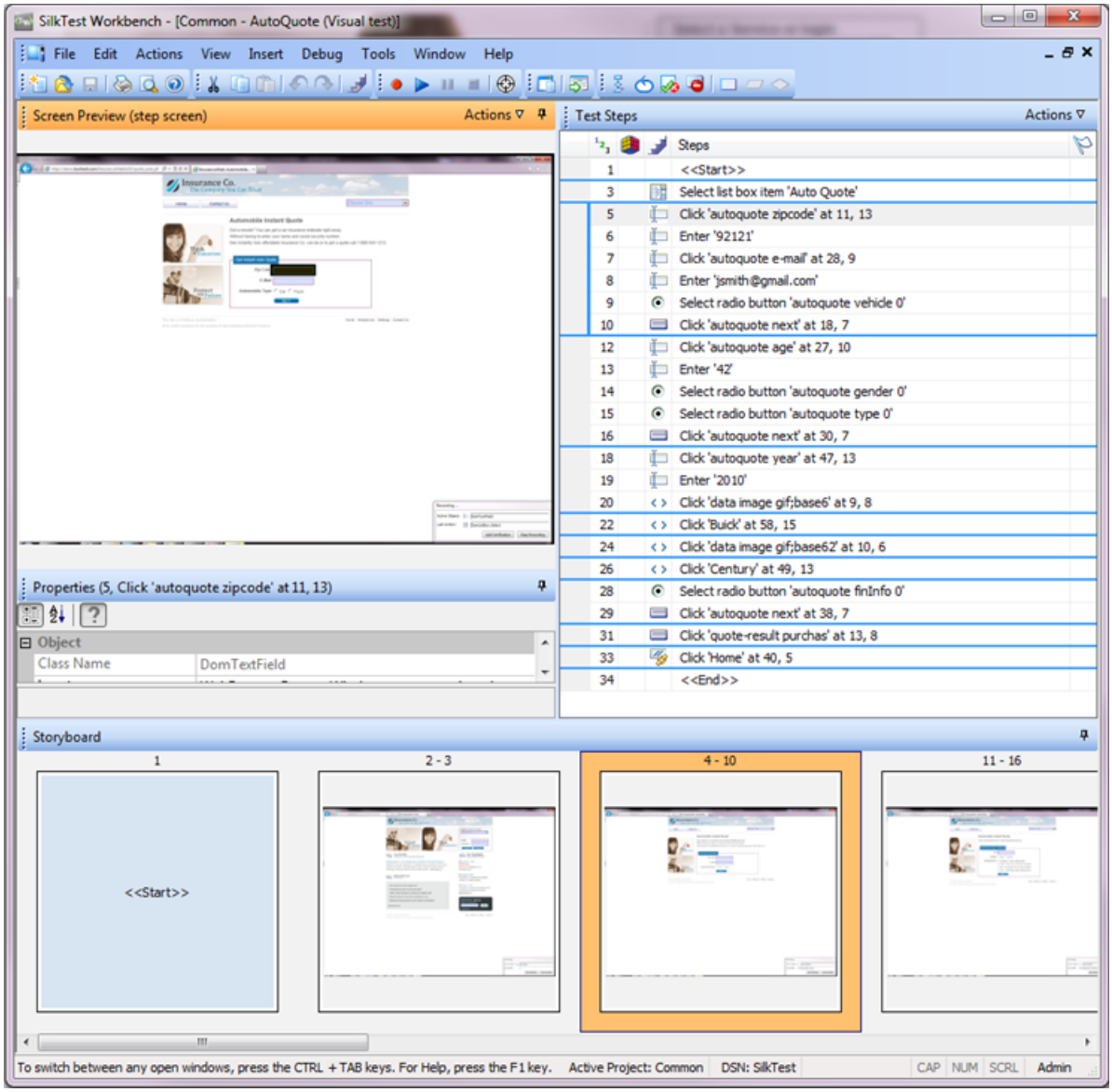
   Because the test has not been named yet, the **Save As** dialog box opens.
2. In the **Name** text box, change the name to `AutoQuote`.
3. In the **Description** text box, type `Visual test tutorial`.
4. Click **OK**. The visual test displays in the Visual Navigator.

Reviewing the Recorded Test Steps

Once you have recorded your visual test and saved it, the visual test displays in the Visual Navigator. The four panes of the Visual Navigator provide a comprehensive view of the visual test.

Steps in the Test Steps pane represent the screens accessed during recording and the actions performed while completing the quote in the Web application.

Your recorded steps should be similar to the steps in the following graphic.



The columns in the Test Steps pane include:

Column	Description
(Number)*	Represents the sequential order in which steps are recorded and played back. Steps display in this sequential order by default. Any steps that have breakpoints added for debugging display a breakpoint in the column next to this column. If you change the view from the default view (Steps and Screens) to the Screens only or the Steps only view, the numbering scheme sequentially skips the steps or screens to indicate the gap in the recording sequence.
(Logic)*	Displays icons representing the type of logic for the step, if the step contains logic.
(Step Type)*	Displays icons representing the type of action being executed in the step.
Steps	Describes the action being taken for the step.
(Flag)*	Displays the **Assigned Flag** icon.

Column	Description
	The **Assigned Flag** icon indicates that the associated step is flagged to appear in the **Test Steps** pane and optionally on the **Start Screen** of the assigned user. Additionally, you can move your pointer over the icon to display a ToolTip containing the flag description, modified date, and assigned date.
(Step Description)*	Displays a user-defined step description. This column does not display in the default view, but can be shown by clicking **Actions** from the **Test Steps** pane title bar and then **View** > **Step Description**. To create a step description, select a step. In the **Properties** pane, update the **Step description** property. A **Step Description** icon appears in this column indicating that the step has a description. Either move your pointer over the icon to display a ToolTip containing the description or select the step and read the description in the **Properties** pane.

Tip: Different panes in the Visual Navigator are synchronized with the **Test Steps** pane. In the preceding graphic, the recorded step that selects **Auto Quote** from the list box is selected in the **Test Steps** pane. As a result:

- The **Screen Preview** shows the state of the application before **Auto Quote** is selected.
- The **Properties** pane shows the properties for the selected step.
- The thumbnail representing the group of steps related to selecting the **Auto Quote** list item is highlighted in the Storyboard.

Scroll through the steps in the visual test and select various steps to view the updated information in the other panes.

Playing Back the Recorded Visual Test

Once you have recorded and saved your visual test, you can play it back to verify that the visual test works.

1. Perform one of the following steps:
   - Choose **Actions** > **Playback**.
   - Click **Playback** on the toolbar.

   The **Playback** dialog box opens. This dialog box lets you determine how the result is saved.
2. In the **Result description** text box, type `Initial test results for the recorded test`.
3. Click **OK**.
4. If multiple browsers that are supported for replay are installed on the machine, the **Select Browser** dialog box opens. Select the browser and click **Run**.

Each result is identified with a unique test run number.

Silk Test Workbench minimizes and the visual test plays back. During playback, the actions you performed while recording the visual test are played back on the screen against the sample application. When playback completes successfully, the **Playback Complete** dialog box opens.

Analyzing Results: Introduction

After playing back a visual test, Silk Test Workbench generates a test result. A test result contains information about the playback of the visual test. Information such as the name of the visual test, the run number, the date and time each step executed, the pass or fail status of each step, and other important information.

Using the Result Window Tabs

To quickly view test result information, the **Result** window contains five tabs which function as filters that organize and display specific types of result information.

1. Click **Go to Result** on the **Playback Complete** dialog box.

   The **Result** window opens to the **Summary** tab by default. The **Summary** tab provides an overview of the test run including information such as whether the playback was successful or not, the latest run number, the number of verifications that passed or failed, the start time and end time of the test, and other basic information about the result of the test run.
2. Click the **Details** tab.

   The **Details** tab displays the result of every step using the four panes of the Visual Navigator:

Test Steps	Lists information about the playback result of each step in the visual test.
Screen Preview	Displays the Web application screens captured during playback.
Properties	Displays the properties of a step.
Storyboard	Provides a graphical outline and overview of a result.

   Note: The **Passed**, **Failed**, and **Flags** tabs also display result information using the Visual Navigator. The only difference is that these tabs display specific types of steps, whereas, the **Details** tab displays every step.
3. Select any step.

   Silk Test Workbench updates the other panes with information specific to the selected step. In the **Screen Preview**, the screen captured during playback is compared against the screen captured when the visual test was first recorded. In the **Properties** pane, the properties of the selected step are listed. And in the **Storyboard**, the group of steps in which the step occurs is highlighted.

   Tip: To view the entire name of the step, you might have to expand the **Steps** column or move your pointer over the step to display a ToolTip containing the entire name.

Using the Result Window Toolbar

The **Result** window toolbar provides several options for customizing the display and type of content found in a result.

1. Click **Advanced View** on the Results toolbar. Silk Test Workbench displays additional columns in the **Test Steps** pane and additional properties in the **Properties** pane.
2. In the **Test Steps** pane, scroll to the right to view the additional columns.

   These columns provide specific information about each step such as, the time in milliseconds it took for the step to run, the user name of the person who ran the test, and other specific details.
3. In the **Properties** pane, the Result property category lists the corresponding properties of each column in the **Test Steps** pane. Scroll down to view the entire list.
4. Click the **View** drop-down arrow and select **Steps Only**. Silk Test Workbench filters out all the screens, so you can quickly see only the steps.
5. Click the **View** drop-down arrow and select **Steps and Screens**. Silk Test Workbench displays all the screens and steps of the result.

   The **Result** window toolbar contains additional options for customizing the display and type of content found in a result.

Using the Properties Pane

The **Properties** pane displays properties of a step that describe the basic characteristics of the step including the name of the step, the execution status, the details about the playback performance, and other information.

1. Click the step `Enter '42'`.

   This step performs the action of typing the value '42' into the **Age** text box.

   The **Properties** pane updates and displays the properties of the selected step.

2. Click the **Categorized** icon if it is not already selected.

   The properties are grouped into the following main categories:

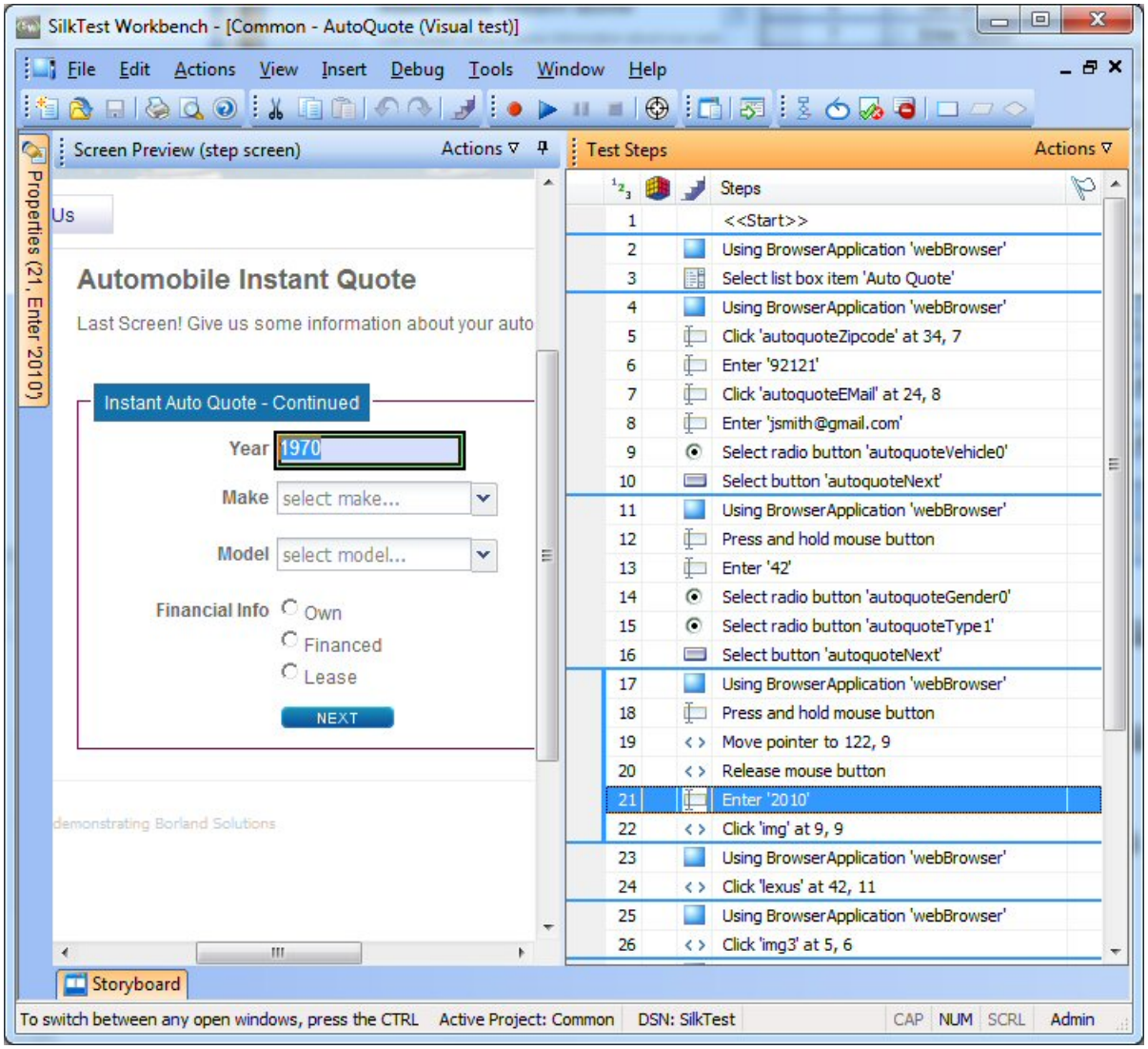
Result	Contains properties that correspond to the columns in the **Test Steps** pane. Properties such as the name of the step, the date and time the step was executed, and the time it took to run the step.
Extended properties	Contains additional playback details such as the locator name Silk Test Workbench uses to identify the control or the value of the text entered by an `Enter` action step. Extended properties are helpful to view the contents of variables or expressions when they are used in action steps. For example, an `Enter` step that uses the variable *textVar* as its value displays the contents of *textVar* in the Extended properties category.

3. Expand the **Extended properties** node. The Text property and Locator property appear with their respective values listed. The Text property value is '42', which is the value entered in the **Age** text box for this step.
4. Click the **Show/Hide Step Properties of Visual Test Before Playback** icon. The Visual test details node appears and lists all of the original properties of the step as they exist in the visual test.

Using the Screen Preview

The **Screen Preview** displays a captured image of the test application for each step in the **Test Steps** pane that interacts with a control. The captured image can show the full desktop, the application window, or only the active window. Captured images represent the state of the application before its associated step is executed.

1. In the **Test Steps** pane, select the step that contains the year of the car. Enter `2010`. In the **Screen Preview**, the screen captured during playback appears next to the screen captured during the recording of the visual test. The control for this step, the browser window, is highlighted by a black box.



2. In the **Screen Preview**, click **Actions** > **Show Differences** > **Off**. The Visual test screen closes and the Playback screen opens.
3. Click **Actions** > **Zoom** > **75%**, then use the scroll bars to position the page so that it displays the year clearly.
4. Switch to the visual test. In the **Test Steps** pane, click **Actions** > **Visual test Window**.

Enhancing the Visual Test: Introduction

Enhancing a visual test includes making updates to the existing visual test to ensure that it works with newer versions of the test application. For example, to handle and verify varying conditions in the test application you can insert test logic. Additionally, to increase the readability of a visual test or to remind yourself or others about important aspects of the test, you can insert a flag, message box, or step description.

These are just some of the ways in which you can use Silk Test Workbench to enhance existing visual test to create more powerful, robust, and flexible tests.

Updating From the Screen Preview

When Silk Test Workbench records a visual test, it captures screens from the test application in addition to the associated controls for each screen. In the **Screen Preview**, Silk Test Workbench displays each captured screen and highlights the control that is identified by a given step in the visual test. From the

Screen Preview, you can update a step by identifying a different control in the captured screen without accessing the test application.

In this lesson, you will select a different button from the captured screen in the **Screen Preview** using Silk Test Workbench's **Insert Control From Screen Preview** feature.

1. In the **Test Steps** pane of the AutoQuote visual test, select the step before the <<End>> step.

   The step text is similar to the following: `Click 'Home' at 40, 5`.

   The **Screen Preview** displays the **Purchase A Quote** page and highlights the **Home** button.
2. In the **Screen Preview**, click **Actions** > **Zoom** > **75%**, then use the scroll bars to position the screen so that the **Contact Us** button is clearly visible.
3. In the **Properties** pane, click the **Locator** text box. The locator selection buttons display in the value area of the locator.
4. In the **Properties** pane, click **Identify from the screen preview**.

   Note: If the button is disabled, you have not enabled browser control capturing before recording the visual test.

   The pointer moves to the **Screen Preview**.
5. Move the pointer over the **Contact Us** button and then click the **Contact Us** button.

   When you click the button, the **Locator** property in the **Properties** pane changes to display the new locator name for the button, and the step text in the **Test Steps** pane changes to the following: `Click 'Contact Us' at 40,5`.
6. Click **Save**.

The next time you play back the visual test, Silk Test Workbench clicks the **Contact Us** button instead of the **Home** button.

Inserting a Verification

A verification is test logic that evaluates a user-defined condition, and then sends a pass/fail message and, optionally, a flag to the playback result of a visual test.

In this lesson, you will insert a verification to ensure that the quote uses the correct vehicle model.

1. Ensure that you are viewing both steps and screens by clicking **View** > **Test Steps** > **Steps and Screens**.
2. Select the step that reads `click 'rx400' at 91,11.`
3. In the **Screen Preview**, click **Actions** > **Zoom** > **75%**, then use the scroll bars to position the screen so that the **Model** is clearly visible.
4. Perform one of the following steps:

   - Click **Create Verification Type Logic** on the toolbar.
   - Choose **Insert** > **Test Logic** > **Verification**.

   The **Welcome** page of the **Test Logic Designer** wizard opens.
5. Click **Next**. The **Select a Logic Type** page opens.
6. Click **The property of a control**, and then click **Next**. The **Define a property-based condition** page opens.
7. Click **Identify from the screen preview**. The pointer moves to the **Screen Preview**.
8. Select the **Model** combo box. The **Define a property-based condition** page displays the name of the selected control and the properties of the control. The control name displays as:

   ```
   webBrowser.browserWindow.modelCombo
   ```
9. In the **Select a property** grid, scroll to **Text** and select it.

10. Ensure that **Select the condition** is set to **Is Equat to**.
11. Change the **Expected value** to use the updated `Text` property by typing **RX400** in the **Expected value** text box.
12. Click **Next**. The **Build the Verification** page opens. From this page, you can define the pass or fail message to send to the playback result. At the top of the page, the condition that Silk Test Workbench verifies should be as follows:

    ```
    If "webBrowser.browserWindow.modelCombo"."Text" Is Equal to "RX400"
    ```

    This condition defines the logic for the verification. The condition compares the model type to the type selected.
13. Replace the default pass text description to `The model type is correct`, and the default failed text description to `The model type is NOT correct`.
14. Click **Next**. The **Summary** page opens.

    A verification step is inserted after the selected step. The step text for the verification step is similar to the following:

    ```
    Verify "webBrowser.browserWindow.modelCombo"."Text" Is Equal to "RX400"
    ```
15. After reviewing the verification, click **Finish**.

You have successfully enhanced the recorded visual test by inserting test logic that verifies the value of a property in the sample application.

Creating a Local Variable to Store Application Data

Variables enhance visual tests by providing the ability to store data values for use in other parts of the test or in other visual tests or scripts. Data can also be output to other types of files.

In this lesson, you will store variable text in the control that displays the email address so it can be used in a later lesson of the tutorial. To do this, you must first create a local variable to store the text.

1. In the **Test Steps** pane, click **Actions** > **Insert** > **Variable** > **Add Local**. The **Add Local Variable** dialog box opens.
2. In the **Variable name** text box, type `strEmailAddress`.
3. From the **Type** list, select **Text**.

   Leave the **Initial value** text box empty for this lesson, since a value will be stored to the variable in a subsequent lesson.

   The **Text** type stores the variable value as a text, or string data type.
4. Click **OK**. The new variable is saved for the visual test. Once the variable is created, you can see it and edit its definition from the `<<Start>>` step.
5. To view the *strEmailAddress* variable, select the `<<Start>>` step in the **Test Steps** pane.

   Tip: The `<<Start>>` step is always the first step in any visual test.

   Properties for the step display in the **Properties** pane. With the `<<Start>>` step selected, *strEmailAddress* displays in the **Variables** category defined as a **Text** variable.

The value area for the variable types indicate the number of variables of the type that are currently defined. In this lesson, one Text variable has been created, so the value area for the item shows that one item is associated with this test.

Now that you have created the local variable to store the quote email address, store the email address text from the application to the variable.

Storing Application Data to the Local Variable

The sample application displays a unique email address on the **Get Instant Auto Quote page** page. The text on this page containing the email address is the property value of a control on the page.

In this lesson, you will store this text to the local variable *strEmailAddress* created in the previous exercise.

1. In the **Test Steps** pane, select the step that shows the email address value.

   The step text should look similar to the following: `Enter 'jsmith@gmail.com'`

   When the step is selected, the captured screen for the **Get Instant Auto Quote page** page displays in the **Screen Preview**.
2. In the **Test Steps** pane, click **Actions** > **Insert** > **Property from Control**.

   This inserts a step into the visual test just after the selected step. The step text should be similar to the following: `Get the '"' property of the control`

   This step will be used to store the email address text from the **Get Instant Auto Quote** page to the local variable by editing the step properties. You will edit the step properties to specify the control to be used, the property of the control, and the variable to store the property value. The variable is *strEmailAddress*, which you created in the previous exercise.
3. In the **Screen Preview**, click **Actions** > **Zoom** > **50%**, and then use the scroll bars to position the screen so that the email address text is clearly visible.
4. In the **Properties** pane, click the **Locator** text box. The locator selection buttons display in the value area of the locator.
5. Click **Identify from the screen preview**. The pointer moves to the **Screen Preview**.
6. Move the pointer over the **E-Mail** text on the page.

   Ensure the highlighted box appears around the text on the page, then click the highlighted area to identify the control.

   Silk Test Workbench updates the **Locator** information in the **Properties** pane with the following value: webBrowser.browserWindow.autoquoteEMail.
7. In the **Properties** pane, click the **Property** text box, and then select **Text** from the list for its value.
8. Click **(Select a local variable...)** and then select **strEmailAddress** from the list.

   Now that you have successfully identified the control, the property of the control containing the desired value, and the variable in which to store the property value, the step text should read as follows: `Put the 'Text' property of the control into variable 'strEmailAddress'`

   The **Local variable name** value in both the **Properties** pane and the step text changes to *strEmailAddress*.

To confirm that the test captures the property value and stores it properly, play back the test and review the result.

Playing Back and Analyzing the Enhanced Visual Test

Now that you have made several enhancements to the recorded test, play back the visual test and analyze the result.

1. Perform one of the following steps:

   - Choose **Actions** > **Playback**.
   - Click **Playback** on the toolbar.

   The **Playback** dialog box opens.
2. In the **Result description** text box, type `Enhanced test results for the recorded visual test`.
3. Click **OK**.
4. If multiple browsers that are supported for replay are installed on the machine, the **Select Browser** dialog box opens. Select the browser and click **Run**. Silk Test Workbench plays back the enhanced test.
5. Click **Go to Result** on the **Playback Complete** dialog box.

   The **Result** window opens to the **Summary** tab by default.

The **Summary** tab shows that the visual test passed, which means it played back successfully without any errors, or failed verifications.

6. Click the **Passed (1)** tab.

   The number in parentheses indicates the total number of verifications that passed. The **Test Steps** pane displays the verification step and the **Result Detail** column displays the pass text description of the verification.
7. Click the **Details** tab to display the result of every step.
8. In the **Test Steps** pane of the **Result** window, scroll down to the step that shows the result of storing the email address to a variable. The step text is similar to the following: `Put the 'Text' property of the control into variable 'strEmailAddress'`.
9. The contents of the *strEmailAddress* variable displays in the **Result Detail** column. To view the entire contents of the **Text** property value, move your cursor over the **Result Detail** column for the step. A ToolTip appears displaying the entire contents of the property.

Congratulations! You have successfully created a visual test that reliably tests the sample application. In the next lesson, you will learn about several advanced testing concepts and features such as how to quickly and easily execute a visual test within another visual test.

Executing a Visual Test Within a Visual Test: Introduction

In this tutorial, you created a single visual test that performs every action required to receive an auto insurance quote from the web application. A single visual test is useful when implementing a basic test case against a simple application. However, most software testing requires a more rigorous approach that involves testing every aspect of an application. An additional requirement is the ability to rapidly update existing visual tests whenever the test application changes.

To provide an efficient means for solving these testing challenges, Silk Test Workbench supports modular testing, in which you can "chunk" common sets of actions of a particular testing solution into a single test, and then reuse the visual test in other visual tests that require the same set of actions.

Modular Testing

Before creating visual tests, scripts, and other Silk Test Workbench assets to build application testing solutions, it is a good practice to plan a testing strategy.

It is not necessary to include all the parts of a specific test solution in a single visual test or script and is not usually beneficial to do so.

Typically, the most efficient testing approach is a modular approach. Think of your application testing in terms of distinct series of transaction units.

For example, testing an online ordering system might include the following distinct transaction units:

- Log on to the online system
- Create a customer profile
- Place orders
- Log off the online system

If one test is created to handle all of these distinct units and there are ten different scenarios that use this test, you would need to record ten different tests to handle the scenarios. If any change occurs to the application, for example if an extra field is added to the logon window, ten different tests would require a change to accommodate data input to the new field.

Rather than creating one visual test or script that tests all of these transaction units, and then recreating it ten times for each scenario, it may be more beneficial to create separate tests as test "modules" that handle each one of these transaction units. If a separate test is created for each of the separate transaction units and reused for each of the test scenarios, then only the test that handles the logon transaction unit would require change.

Now that you understand the basics of modular testing, you are ready to create a second test and add it to the test you created in the previous lessons.

Recording the Second Visual Test

In this section of the lesson, you record a second visual test for the tutorial and learn an alternate way to create a visual test asset.

1. Choose **File** > **New**. The **New Asset** dialog box opens.
2. Select **Visual test** from the asset types list, and then type a name for the visual test asset in the **Asset name** text box.
   For this tutorial, type `AddAccount` for the name.
3. Check the **Begin Recording** check box to start recording immediately.
4. Click **OK** to save the visual test as an asset and begin recording. The **Configure Test** dialog box opens.
5. Select the **Web** tab.
6. Select **Internet Explorer** from the list.
7. From the **Home** page of the sample application, click **Sign Up** in the **Login** section. The **Create A New Account** page opens.
8. Provide the following information in the appropriate fields.

   Press the `Tab` key to move from one field to the next.

Field Name	Value
First Name	Pat
Last Name	Smith
Birthday	February 12, 1990 **Note:** Click the down arrow next to the month and year in the calendar control to change the month and year and then select 12 on the calendar.
E-Mail Address	smith@test.com
Mailing Address	1212 Test Way
City	San Diego
State	CA
Postal Code	92121
Password	test

9. Click **Sign Up**.
10. Click **Continue**. The contact information is displayed.
11. Click **Home** near the top of the page to return to the home page where recording started.
12. Click **Log Out**.
13. Press `Alt+F10` to complete recording. The **Recording Complete** dialog box opens.
14. Click **Save**. The visual test opens in the Visual Navigator.

Inserting One Visual Test Within Another

In this section of the lesson, you will learn how to insert the second visual test, which adds a user account, in the original visual test before the steps that perform the request for an auto quote.

Executing visual tests within visual tests is a powerful method for efficiently testing the same basic steps in multiple visual tests.

Tip: When inserting a visual test within another visual test, it is important to ensure that any test applications are in the correct initial playback state.

1. From the **Recent** list in the **Start Screen**, click `AutoQuote` to open it.

   `AutoQuote` is the first visual test that you created in this tutorial.
2. In the **Test Steps** pane, select the `<<Start>>` step.
3. Click **Actions** > **Insert** > **Visual test**. The **Browse for Visual test** dialog box opens.
4. From the **Select an asset** list, select the visual test named **AddAccount** and then click **OK**.

   Tip: If there are many visual tests in the **Select an asset** list, type **AddAccount** into the **Name Filter** text field and then click on the visual test.

   Silk Test Workbench inserts a step before the selected step. The inserted step calls the selected visual test. The step text is as follows:

   ```
   Playback visual test 'AddAccount'
   ```

   Tip: During playback, when the preceding step executes, the original visual test plays back to completion before the inserted visual test plays back.

In the next lesson, you will learn how to playback this modular visual test, and respond to a playback error.

Responding to Playback Errors: Introduction

Errors encountered during playback can be caused by a variety of factors, such as changes in the test application and improper visual test step flow. Quickly diagnosing and fixing these errors using the debugging features minimizes visual test maintenance and allows for a more efficient team testing effort.

First, begin this lesson by playing back the modular test you created in the previous lesson.

Playing Back the Modular Test

In the previous lesson, you created a modular test by inserting one visual test, AddAccount, into another visual test, AutoQuote.

In this section of the lesson, you will play back the modular test and encounter an error during playback.

1. With the AutoQuote visual test open, click **Playback** on the toolbar. The **Playback** dialog box opens.
2. In the **Result description** text box, type `Responding to errors in a modular test`.
3. Click **OK**.
4. If multiple supported browsers are installed on the machine, the **Select Browser** dialog box opens. Select the browser and click **Run**.

   During playback, the test stops on the **Create A New Account** page and an error message opens.

   This error occurs because the database requires a unique email address for each customer record. Since you have already entered the email address during the recording of the AddAccount visual test, the email address already exists in the database and the test fails.

Now that you have encountered a playback error, you are ready to debug the test.

Debugging Errors

After Silk Test Workbench encounters a playback error, the Silk Test Workbench **Playback Error** dialog box opens and provides the option to enter Debug mode. In Debug mode, playback is suspended, which allows you to diagnose and fix any playback errors using the Silk Test Workbench debugging features.

In this section of the lesson, you will learn how to debug the error that occurred during playback of the modular test in the previous section.

1. From the **Playback Error** dialog box, click **Debug**.

   In Debug mode, playback is suspended. This allows you to fix the error by either editing the properties of the step incurring the error, deleting the step, disabling the step, or copying and pasting a step from another visual test before resuming playback.

Silk Test Workbench enters Debug mode and displays the AddAccount visual test with the step incurring the error highlighted in yellow.

2. Choose **Edit** > **Enable/Disable** to disable the step `Select button 'id=signup:continue'`.

   By disabling this step, you prevent the error from occurring the next time you play back the test.

   The step text turns grey and italicized indicating that it is disabled.
3. Click **Playback** on the toolbar. The **Playback Error** dialog box opens.
4. Click **Debug**.
5. Choose **Edit** > **Enable/Disable** to disable the step `Select button 'id=logout-form:logout'`.

   This error occurs because the test is looking for the **Log Out** button on the Home page, but since the Sign Up Continue step is disabled, the button does not display on the page. We will fix this error later in this tutorial.
6. Choose **Debug** from the menu.

   The following Step commands that appear at the top of the menu allow you to control the step execution:

Step Into (F8)	Executes playback one step at a time. **Step Into** is useful to trace through each step, and steps into other visual tests or scripts that are inserted into the visual test being played back. Each inserted visual test or script is also executed one step or line at a time. **Step Into** is useful for detailed analysis of a test, and lets you see the effect of each step on variable usage and test application interaction.
Step Over (`Shift + F8`)	Executes an entire visual test or script inserted into another visual test as if it were a single step. Use **Step Over** when playback is in debug mode for a step that plays back a visual test or script. This plays back the inserted visual test or script in its entirety. Once the inserted visual test or script plays back in its entirety, playback suspends in debug mode at the next step in the original visual test. Using **Step Over** at a step other than one that plays back another visual test or script has the same effect as using **Step Into**. Only the next step executes before playback suspends and re-enters debug mode.
Step Out (`Ctrl + Shift + F8`)	Executes all remaining steps in a visual test or script being played back from another visual test, then suspends playback at the next step in the original visual test. Use **Step Out** when playback is suspended at a step in a visual test or code line that has been inserted in another visual test, and you want to playback the remaining visual test or script and return to the original visual test. When playback executes the remainder of the inserted script or visual test, it suspends and re-enters debug mode at the next sequential line in the original visual test.
Run To Cursor (`Ctrl + F8`)	Allows you to select a step where you want playback suspended. This allows you to "step over" selected sections of a visual test or script. Use **Run To Cursor** to playback the visual test or script and stop playback at a point just before a run-time error occurs. This lets you stop playback at a specific line or statement without having to insert breakpoints. Once playback stops, you can continue using one of the other debug options.
Run From Cursor	Plays back the visual test from the currently selected test step.

7. Choose **Step Out**.

   This command executes the remaining steps in the AddAccount visual test, and then suspends playback in the next step of the AutoQuote visual test.

   After selecting **Step Out**, Silk Test Workbench displays the home page, which is the last step in AddAccount, and suspends playback. Silk Test Workbench then opens AutoQuote in Debug mode and highlights the next step.

Next, you learn how to monitor the values of variables used in the visual test.

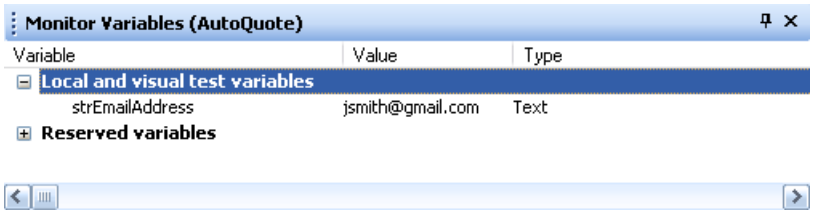
Tracking Variables During Playback

In Debug mode, you can track the playback values of both local and reserved variables using the Local Variables window.

In this section of the lesson, you will learn how to use the Local Variables window to view the value of the local variable you created in a previous lesson.

1. Choose **Debug** > **Local Variables**. The Local Variables window opens.
2. Expand the **Local and visual test variables** node.

   The local variable you created in a previous lesson, *strEmailAddress*, appears. During playback, the value for this variable will appear in the **Value** column once the step using the variable executes. The result looks like the following graphic:



   Note: If you want to see the variable values in the Local Variables window but your test does not have any errors, insert a breakpoint in the visual test. To insert a breakpoint, select the step following the variable step, choose **Debug** > **Set/Clear Breakpoint** and then playback the test.
3. To see how the **Run to Cursor** debugging works, select the step `Click 'Lexus' at 24,9` and then choose **Debug** > **Run to Cursor**. Silk Test Workbench plays back the remaining steps until the step `Click 'Lexus' at 24,9`, and then displays the AutoQuote visual test.

Now you are ready to complete the playback of the modular test and review the test results.

Reviewing the Result

Review the results of the visual test after you have finished debugging the visual test.

1. Click **Playback** on the toolbar to complete playback of the AutoQuote visual test. The **Playback Complete** dialog box opens.
2. Click **Go to Result**. The AutoQuote result appears with the **Summary** tab displayed by default.

   The **Summary** tab displays the overall details of the test run. Note that the **Visual tests or .NET Scripts (number of times each ran)** field lists AutoQuote(1) and the inserted visual test, AddAccount(1).
3. Click the **Details** tab.
4. In the **Test Steps** pane, scroll down to the steps in blue text.

   By reviewing the **Result** and **Result Detail** columns, you can quickly find information about any errors that occurred during playback.

   Note: To view all steps in the **Test Steps** pane, make sure you have Steps and Screen selected. Click **Actions** > **View** > **Steps and Screens**. The **Failed** tab does not display steps containing playback errors. It only displays failed verifications.

You have learned how to diagnose and debug playback errors.

Using ActiveData

To effectively mimic application use, application testing often involves performing the same action or set of actions repeatedly using different sets of data. For example, the previous lesson included recording a test

that created a customer record. To create ten customer records, you can record ten different tests, each with its own set of customer data. However, with Silk Test Workbench you can enhance this original test to run repeatedly for ten iterations and use a different set of data for each iteration.

With ActiveData testing, you can use data in external files as input to the test application, and then repeat selected steps using different data for each iteration.

In this lesson, you will learn how to:

- Create an ActiveData asset and associate it with a visual test
- Create repetition logic that repeats selected steps a specified number of times, using different data for each repetition
- Define how to use the ActiveData file in the visual test
- Define the steps to be repeated
- Map ActiveData in a data file to literal data in a visual test

In the previous lesson, you recorded a visual test that entered customer information for Pat Smith into the customer database. Each time that visual test plays back, Silk Test Workbench uses the literal data values that were captured during the initial recording.

In this lesson, you replace the literal data values used to input customer information for Pat Smith with ActiveData, so that when Silk Test Workbench plays back the visual test, different customers contained in the external file are used.

Before creating ActiveData for the visual test, review the ActiveData file.

Reviewing the ActiveData File

When creating an ActiveData asset, you can either select an existing data file or create a new file to contain the data used by the asset. In this tutorial, you will use an existing file named customers.csv, a comma-separated file containing customer information. This file is located in the Examples folder of your Silk Test Workbench installation directory.

Each column of the customers.csv file corresponds to a field used to enter a customer into the database, with the exception of the Age column, which will be used in a later lesson in this tutorial.

When using ActiveData for this test, Silk Test Workbench plays back the test and repeats the steps that enter customer information, each time using data for a different customer, until each customer in the file is entered into the database.

Tip: In the ActiveData file, password values are encoded text.

The ActiveData file contains the following columns:

- First Name
- Last Name
- Birthday
- Email Address
- Mailing Address
- City
- State
- Postal Code
- Password
- Age (used in a subsequent lesson)

Next, create the ActiveData test asset and associate it with the visual test.

Creating the ActiveData Test Asset

Before using a data file in ActiveData testing, you must create an ActiveData asset that uses the file, and then associate the ActiveData asset with the visual test.

1. In the **Recent** list on the Start Screen, double-click the visual test named **AddAccount** to open it. The AddAccount visual test opens in the Visual Navigator.
2. In the **Test Steps** pane, click **Actions** > **Insert** > **ActiveData** > **New**. The **ActiveData asset setup** window opens.
3. In the **Name** text box, name the ActiveData asset by typing `customers`.
4. Click **Browse** to search for the `customers.csv` file to associate with the AddAccount ActiveData asset. The **Choose ActiveData Asset** dialog box opens.
5. Navigate to the sample file location and select the sample ActiveData file, `customers.csv`.

   By default, the location is: `C:\Program Files\Silk\Silk Test\examples\customers.csv`.
6. Click **Open**. The path and file name appear in the **File** text box of the **General** tab.
7. In the **ActiveData asset setup** window, click the **Options** tab.
8. Check the **Use first row as header** check box.

   Tip: Click the **Details** tab to view the contents of the ActiveData file.

   This setting treats the first row of data in the ActiveData file as a header row, not as data.
9. Click **Save and close** to create the ActiveData asset and associate it with the AddAccount visual test.

In a visual test, information about any associated ActiveData assets is stored in the **Properties** pane of the `<<Start>>` step. To review each ActiveData asset associated with the visual test , select the `<<Start>>` step and review the **ActiveData** property in the **Properties** pane.

Now that the ActiveData asset has been created and associated with the visual test, you are ready to create the repetition logic that will use the active data in the test.

Creating Repetition Logic for ActiveData Files

In visual tests, ActiveData typically involves repeating a series of steps, and substituting literal data for ActiveData for each repetition.

In this lesson, we will substitute the literal data that adds a customer to the database with data from the ActiveData file customers.csv, which contains ten customer records. For each repetition, data from a customer record in the ActiveData file will be used to enter data in the fields on the **Create a New Account** page of the InsuranceCo Web site.

1. Make sure the AddAccount visual test displays in the Visual Navigator.
2. Click **Create Repetition Type Logic** on the Silk Test Workbench toolbar. The **Test Logic Designer** wizard appears with the **Welcome** page displayed.
3. Click **Next**. The **Select a Logic Type** page opens.
4. Click **Repeating a sequence of steps using data from an ActiveData file**, and then click **Next**. The **Define the ActiveData asset to use** page opens.
5. Since only one ActiveData asset (customers) has been associated with the visual test, it should appear in the **ActiveData asset** list. If not, select it from the list.
6. *Optional:* From the **Sheet name** list, select the sheet to be used from the data file.

   By default, the sheet that is specified in the ActiveData asset is used.
7. In the **Start row** text box, leave the default value at **1**.
8. Check the **End at last row containing data** check box.
9. Click **Get all rows in sequence**.

   You have now defined how the data file will be used in the repetition logic, and how many repetitions will occur.

   - The first row of the Active data file will be used first when the repetition logic executes.
   - The repeat will continue using each row until the last row is used.

- Since all rows of data will be used in the repetition logic in sequential order and there are 10 rows of data in the ActiveData file, the steps to add a customer to the database will be repeated 10 times. For each repetition, a different row of customer data from the ActiveData file will be used as input.

10. Click **Next**. The **Build the Repeat** page opens.

Now that you have defined how the ActiveData will be used, you are ready to determine the steps in the visual test that will be repeated.

Defining the Steps to Repeat

When determining the steps in repetition logic to repeat, consider all the actions that require repetition, not just the steps where literal data is substituted.

For this lesson, there are several steps where literal data will be substituted, but the entire process of accessing the **Create a New Account** page of the InsuranceCo Web site must be repeated in order for each customer in the ActiveData file to be entered. Therefore, all steps in the actual test process must also be repeated, including steps that access the page and return the test to the **Home** page.

1. In the **Build the Repeat** page, click the **Do step** link. The **Select Steps** dialog box opens.
2. Select the step that begins interaction with the InsuranceCo Web site. This is the step immediately after the `<<Start>>` step.

   Because the visual test contains all the steps required in the process of adding a customer, all the steps must be repeated.
3. Click **OK**. The **Select Steps** dialog box closes and the text for the selected step appears in the **Do step** link.
4. In the **Build the Repeat** page, click the **To step** link. The **Select Steps** dialog box opens.
5. Select the step that ends interaction with the InsuranceCo Web site. This is the step immediately before the `<<End>>` step.
6. Click **OK**. The **Select Steps** dialog box closes and the text for the selected step appears in the **To step** link.
7. Click **Next**. The **Summary** page of the **Test Logic Designer** opens. This page displays the test logic you have defined for the visual test.
8. Click **Finish** to insert the test logic into the test steps of your visual test.

   In the visual test, Silk Test Workbench inserts a step after the `<<Start>>` step that starts the repetition logic. The step text should be as follows:`Repeat using activedata 'customers'`

   Silk Test Workbench also inserts a step before the `<<End>>` step that ends the repetition logic. The step text should be as follows:

   ```
   End Repeat
   ```

   All steps between the newly inserted repetition logic steps now appear as indented steps nested between the repetition logic steps. The final part of using ActiveData in the visual test is to map data in the ActiveData file to the literal data in the visual test.

Mapping ActiveData to Literal Data

Before a visual test can use data in an ActiveData file, data in the applicable test steps must be mapped to use data in columns of an ActiveData file.

1. Select the first step in the visual test that enters data into a field.

   This should be the step that enters 'Pat' into the **First Name** text box on the **Create a New Account** page of the InsuranceCo Web site, as originally recorded in the visual test.

   Properties for the step appear in the **Properties** pane.
2. In the **Properties** pane, in the **Parameters** category, select the **text** parameter, which indicates the text that was set to **Pat**.

3. Click **Select** in the value area and click **ActiveData**.

   Tip: If **customers** is not in the **ActiveData asset** list, it has not been associated with the visual test.

   The **Select an ActiveData Column** dialog box opens. Use this dialog box to map the data from the selected property in the test step to data from a column in the ActiveData file.

   The **ActiveData asset** text box shows the ActiveData asset **customers**, which was associated with the visual test. The column names from the customers ActiveData file display in the **Columns** list.

4. In the **Columns** list, select the **First Name** column.

   This is the column in the file that contains first names, which you are mapping to substitute for the literal data 'Pat'.

5. Click **OK** to close the dialog box and map the data.

   Silk Test Workbench replaces the actual data in the test step with an expression that maps to the selected data in the ActiveData file. The existing step text `Enter 'Pat'` now shows the name of the ActiveData asset, the type of data being substituted, and the column name:

   ```
   Enter '[[customers].Text("First Name")]'
   ```

6. Repeat steps 1 through 5 for all other test steps containing data to be substituted with data in an ActiveData file associated with the visual test. Start with the next step, which is the step containing the step text `Enter 'Smith'` text.

   The following table shows:

   - The text of each step in this visual test whose literal date is to be substituted with data from the ActiveData file
   - The name of the step property whose data is to be substituted
   - The ActiveData column to select that contains the data that will be used in place of the literal data

   Note: Steps updated using ActiveData that enter text into a password protected text box do not show ActiveData information in the step text.

Step Text	Property	ActiveData Column
Enter 'Smith'	Text	Last Name
Enter 'smith@test.com'	Text	Email Address
Enter '1212 Test Way'	Text	Mailing Address
Enter 'San Diego'	Text	City
Select list box item 'California'	List Box Item	State
Enter '92121'	Text	Postal Code
Enter 'test'	Text	Password

Now that this visual test has been set up to use data from an ActiveData file, play it back and review the results.

Playing Back and Analyzing the ActiveData Visual Test

Now that the visual test that adds a customer has been enhanced to use customer records from an ActiveData file, it can be played back. When the visual test plays back, the customer records from the ActiveData file will actually be entered into the customer database.

1. Perform one of the following steps:

   - Choose **Actions** > **Playback**.
   - Click **Playback** on the toolbar.

   The **Playback** dialog box opens.

2. In the **Result description** text box, type `AddCustomers using ActiveData`.
3. Click **OK**.
4. If multiple browsers that are supported for replay are installed on the machine, the **Select Browser** dialog box opens. Select the browser and click **Run**. Silk Test Workbench plays back the enhanced test.
5. Click **Go to Result** on the **Playback Complete** dialog box.

   The **Result** window opens to the **Summary** tab by default.
6. Click the **Details** tab to display the result of every step.

   The **Details** tab displays the result of every step executed during playback. Use the scrollbar to scroll though the executed steps.

   There are more steps in the **Test Steps** pane of the **Result** window than there are actual steps in the visual test. This is because each repetition includes its own set of executed steps.

Note: You can also use expressions to update the ActiveData in a visual test. The **Expression Designer** enables you to use data from an ActiveData file column as input, and create an expression that changes the value of the data. The updated data can then be saved back to the same column in the same ActiveData file. For details, see *Updating ActiveData in a Visual Test Using an Expression* in the online help.

Playing Back Scripts From Visual Tests

Scripts and visual tests are similar in that both assets automate manual user actions such as selecting menu items and entering data in a test application. The difference between these two assets is how the user actions are represented in the asset. A script uses a scripting language, Microsoft's Visual Basic running in the Microsoft .NET framework, whereas, a visual test uses steps generated by the point-and-click interface of the Visual Navigator.

With either asset you can create powerful and flexible automated tests that run independently of each other. The choice of asset is dependent on your needs and preferences. You can also use each asset in conjunction with the other. For example, you can create a script that performs a specialized task, and then insert a step in a visual test that plays back the script. In this way, you can leverage the power of the scripting language used in scripts to supplement your visual tests.

When scripts are played back from a visual test, they are the equivalent of a function that you can call whenever you need to perform a repetitive and specialized testing task. This approach is helpful in a team testing environment, where an experienced tester can create a library of scripts that perform common testing functions from which a novice developer can select from when creating a visual test.

Creating a Script to Generate Random Numbers

The first step is to create a script that generates a random number that you can use for the age of the user.

1. Choose **File** > **New**. The **New Asset** dialog box opens.
2. Select **.NET Script** from the asset types list, type `function_randomAge` in the **Asset name** text box and click **OK**.

   Tip: By naming your script "function" you can create an organized library of commonly used testing tasks from which you can quickly select and insert into a visual test.

   The script opens in the **Code** window.
3. Place your cursor on the line following `Sub Main()`, and type:

```
Dim rand As New Random()
Dim TSrandomAge As Integer = rand.Next(10000, 99999)
MsgBox(TSrandomAge)
```

4. Click **Playback** on the toolbar. The **Playback** dialog box opens.
5. Click **OK**. A message box opens and displays a randomly generated number between 10,000 and 99,999.

6. Click **OK**. The **Playback Complete** dialog box opens.
7. Click **Go to .NET Script**.
8. Replace the numeric range parameters (10000, 99999) with the parameters *MinVal* and *MaxVal*. You can use these two parameters to set the range of the random number from the visual test that plays back this script instead of having to open the script and change the values. Type the following:

```
Dim TSrandomAge As Integer = rand.Next(MinVal, MaxVal)
```

9. To verify the script works properly, assign the parameters the following values after `Sub Main()`:

```
Dim MinVal=16
Dim MaxVal=105
```

Your script should look like the following code:

```
Public Sub Main()
  Dim rand As New Random()
  Dim MinVal=16
  Dim MaxVal=105
  Dim TSrandomAge As Integer = rand.Next(MinVal, MaxVal)

  MsgBox(TSrandomAge)
End Sub
```

10. Click **Playback** on the toolbar. The **Playback** dialog box opens.
11. Click **OK**. A message box appears and displays a randomly generated number between 16 and 105.
12. Click **OK**. The **Playback Complete** dialog box opens.
13. Click **Go to .NET Script**.
14. To set the *MinVal* and *MaxVal* parameters using input parameters, perform the following steps:
    a) Modify the `Main()` sub to include the input parameters that you want to create.

```
Public Sub Main(args As IDictionary(Of String, Object))
  Dim MinVal=args ("MinVal")
  Dim MaxVal=args ("MaxVal")
```

    where `"MinVal"` and `"MaxVal"` are the names of the input parameters that we will create in the next procedure.
    b) Modify the `Main()` sub to include the output parameters that you want to create.

```
Public Sub Main(args As IDictionary(Of String, Object))

  args ("TSrandomAge")=TSrandomAge
```

    where `"TSrandomAge"` is the names of the output parameters that we will create in the next procedure.
    c) Since the script will pass the age to the visual test, comment out the message box code by placing an apostrophe in front of the statement.

```
'MsgBox(TSrandomAge)
```

The entire script should look like the following:

```
Public Module Main
    Dim _desktop As Desktop = Agent.Desktop

    Public Sub Main(args As IDictionary(Of String, Object))

        Dim rand As New Random()

        Dim MinVal= args("MinVal")
        Dim MaxVal= args("MaxVal")

        Dim TSrandomAge As Integer = rand.Next(MinVal, MaxVal)
        args ("TSrandomAge")=TSrandomAge
        'MsgBox(TSrandomAge)
```

```
    End Sub
End Module
```

Next, define the script input and output parameters used to pass the random age to the visual test.

Defining the Script Input Parameters

Scripts can receive data from a visual test in an input parameter, and, conversely, pass data to visual tests in an output parameter. In this task, you will define two input parameters that set the range for the random number created in the function_randomAge script.

1. In the **Properties** pane, right-click and choose **Add Input Parameter**. The **Add Script Input Parameter** dialog box opens.
2. In the **Name** text box, type **MinVal**.
3. From the **Type** list, select **Number (Double)**.
4. In the **Default Value** text box, type `16`.

   Optionally, you can set the default value in the visual test.
5. Click **OK**. The input parameter displays in the list of input parameters in the **Properties** pane.
6. In the **Properties** pane, right-click and choose **Add Input Parameter**. The **Add Script Input Parameter** dialog box opens.
7. In the **Name** text box, type **MaxVal**.
8. From the **Type** list, select **Number (Double)**.
9. In the **Default Value** text box, type `105`.

   Optionally, you can set the default value in the visual test.
10. Click **OK**. The input parameter displays in the list of input parameters in the **Properties** pane.

Defining the Script Output Parameters

Scripts can receive data from a visual test in an input parameter, and, conversely, pass data to visual tests in an output parameter. In this task, you will define a parameter that passes the random number created in the function_randomAge script to the visual test.

1. In the **Properties** pane, right-click and choose **Add Output Parameter**. The **Add Script Output Parameter** dialog box opens.
2. In the **Name** text box, type **TSrandomAge**.
3. From the **Type** list, select **Number (Double)**.
4. Leave the **Default Value** text box empty.
5. Click **OK**. The output parameter displays in the list of output parameters in the **Properties** pane.

Define a local variable in the visual test to receive the output parameter.

Setting Up the Visual Test to Use Script Data

To set up a visual test to use script data, you must insert a step that plays back the script, create a local variable to store the script data, and associate the local variable with the script output variable. Additionally, to pass data from a visual test to a script, you must set the values of the script input variables.

1. Open the AutoQuote visual test.
2. Choose **File** > **Save As**. The **Save As** dialog box opens.
3. Since you will be modifying the AutoQuote visual test, rename it to **AutoQuote_Modified** to differentiate it from the original test, and then click **OK**.
4. Select the step after the following step text:

   ```
   Playback visual test 'AddAccount'
   ```

5. In the **Test Steps** pane, click **Actions** > **Insert** > **.NET Script**. The **Browse for .NET Script** dialog box opens.
6. Select **function_randomAge**, and then click **OK**. Silk Test Workbench inserts a step that plays back the script.
7. To add a local variable to store the script data, perform the following steps:
   a) In the **Test Steps** pane, click **Actions** > **Insert** > **Variable** > **Add Local**. The **Add Local Variable** dialog box opens.
   b) In the **Variable name** text box, type `VTrandomAge`.
   c) From the **Type** list, select **Number (Double)** and leave the **Initial value** set to 0.
   d) Click **OK**. You can see the variable and edit its definition from the `<<Start>>` step.
8. Select the step for the script.

   The step looks like:`Playback .NET Script 'function_randomAge'`.
9. In the **Properties** pane of the inserted script step, click in the value area of the **Pass contents of 'TSrandomAge' into** property and select the local variable **VTrandomAge**.

   Since we added input parameters in the script, those values display in the **Properties** pane. However, you can modify the input parameters used in the visual test by editing the values. These values are not changed in the script.

Your visual test is now set up to use script data. Each time the AutoQuote_Modified visual test plays back, the function_randomAge script is played back and passes a unique random number to the local variable, VTrandomAge, in the visual test.

Now, modify the step that enters the age information to use the visual test local parameter containing the random number generated in the function_randomAge script.

Using Script Data in the Visual Test

In this section of the lesson, you will modify the step that enters the age information to use the visual test local parameter containing the random number generated in the function_randomAge script.

1. Select the step that enters data in the **Age** text box. The step text is as follows:
   `Enter '42'`
2. In the **Properties** pane, select the **Text** property.
3. Click the **Select** button, and then select **Variable**. The **Select a Variable** dialog box opens.
4. Select **VTrandomAge**, and then click **OK**. The step text displays as:`Enter '[VTrandomAge]'`

This step will now enter the value of the *VTrandomAge* local variable in the **Age** text box.

Playing Back and Reviewing Test Results

1. Perform one of the following steps:

   - Choose **Actions** > **Playback**.
   - Click **Playback** on the toolbar.

   The **Playback** dialog box opens.
2. In the **Result description** text box, type `Adding Customers using data passed from a script`.
3. Click **OK**.
4. If multiple browsers that are supported for replay are installed on the machine, the **Select Browser** dialog box opens. Select the browser and click **Run**. Silk Test Workbench plays back the enhanced test and creates 10 new customers using the random number passed from the script and the additional customer information you defined in the visual test.
5. Click **Go to Result** on the **Playback Complete** dialog box.

The **Result** window opens to the **Summary** tab by default.

6. Click the **Details** tab to display the result of every step.

Silk Test Workbench Script Tutorial

Welcome to the Silk Test Workbench Script tutorial. In this tutorial, you will learn the basic steps required to create a script, play back the script, and then analyze the results of the playback. Additionally, you will learn how to use a number of features that allow you to quickly update and enhance a recorded script.

This tutorial assumes some basic knowledge of Microsoft Visual Basic and the Microsoft .NET framework. If you are unfamiliar with the .NET framework, refer to the Microsoft web site for additional help.

This tutorial uses the Silk Test sample Web application, *http://demo.borland.com/InsuranceWebExtJS/*, to create a real world scenario in which you practice using Silk Test Workbench to create repeatable tests.

Note: The sample application used in this tutorial is designed and optimized to run on Internet Explorer. To ensure a user experience consistent with the lessons in the tutorial, Micro Focus does not recommend running the tutorial sample application on one of the other supported browsers instead of Internet Explorer.

Note: Before you record or playback web applications, disable all browser add-ons that are installed in your system. To disable add-ons in Internet Explorer, click **Tools** > **Internet Options**, click the **Programs** tab, click **Manage add-ons**, select an add-on and then click **Disable**.

The lessons in this tutorial are designed to be completed in sequence as each lesson is based on the output of previous lessons.

Recording a Script: Introduction

As you perform actions to create an insurance quote request in the sample Web application, Silk Test Workbench records the actions. When you have completed recording the actions needed for a script, you can see the recorded script in the **Code** window.

Starting the Sample Web Application

For this tutorial, use the Silk Test sample Web application. This Web application is provided for demonstration purposes.

Use the Silk Test sample Web application with Internet Explorer. To ensure a user experience consistent with the lessons in the tutorial, we do not recommend running the sample Web application with one of the other supported browsers instead of Internet Explorer.

1. To record DOM functions to make your test faster and more reliable, perform the following steps:
   a) Click **Tools** > **Options**.
   b) Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
   c) Click **xBrowser**.
   d) From the **Record native user input** list box, select **No**.
   e) Click **OK**.

   Note: Typically, when you test Web applications, you use native user input rather than DOM functions. Native user input supports plug-ins, such as Flash and Java applets, and applications that use AJAX, while high-level API recordings do not.
2. Before you record or playback Web applications, you must disable all browser add-ons. To ensure that all browser add-ons are disabled, perform the following steps:
   a) In Internet Explorer choose **Tools** > **Internet Options**. The **Internet Options** dialog box opens.
   b) Click the **Programs** tab and then click **Manage add-ons**. The **Manage Add-ons** dialog box opens.
   c) In the list of add-ons, review the **Status** column and ensure that the status for each add-on is **Disabled**.

If the **Status** column shows **Enabled**, select the add-on and then click **Disable**.

d) Click **Close** and then click **OK**.

3. To access the sample application remotely, click *http://demo.borland.com/InsuranceWebExtJS*. The sample application Web page opens.

Recording a Script for the Sample Web Application

During recording, Silk Test Workbench records all interactions with the test application (except interaction with Silk Test Workbench itself) until recording is stopped. After you have finished recording, you can modify the script you have generated to add and remove steps.

1. Choose **File** > **New**. The **New Asset** dialog box opens.
2. Select **.NET Script** from the asset types list, and then type a name for the script in the **Asset name** text box.
   For example, type `AutoQuote` as the title.
3. Check the **Begin Recording** check box to start recording immediately.
4. Click **OK** to save the script as an asset and begin recording. The **Select Application** dialog box opens.
5. Select the **Web** tab.
6. Select **Internet Explorer** from the list.
7. In the **Enter URL to navigate** text box, type `demo.borland.com/InsuranceWebExtJS/`.
8. Click **OK**. Silk Test Workbench minimizes and the **Recording** window opens.
9. In the Insurance Company Web site, perform the following steps:
   a) From the **Select a Service or login** list box, select **Auto Quote**. The **Automobile Instant Quote** page opens.
   b) Type a zip code and email address in the appropriate text boxes, click an automobile type, and then click **Next**.
   For example, type `92121` as the zip code, `jsmith@gmail.com` as the email address and specify `Car` as the automobile type.
   c) Specify an age, click a gender and driving record type, and then click **Next**.
   For example, type `42` as the age, specify the gender as `Male` and `Good` as the driving record type.
   d) Specify a year, make, and model, click the financial info type, and then click **Next**.
   For example, type `2010` as the year, specify `Lexus` and `RX400` as the make and model, and `Lease` as the financial info type.
   A summary of the information you specified appears.
   e) Click **Purchase**.
   The **Purchase A Quote** page opens.
   f) Click **Home** near the top of the page to return to the home page where recording started.
10. Stop recording by pressing `Alt+F10`, clicking **Stop** in the **Recording** window, or clicking the Silk Test Workbench taskbar icon. The **Recording Complete** dialog box opens. If the **Do not show this message again** check box is checked in the **Recording Complete** dialog box, this dialog box does not appear after recording is stopped. In this case, the script displays.
11. Click **Go to .NET Script**. The script displays in the **Code** window.
12. Click **Save**.

Reviewing the Recorded Script

Silk Test Workbench records all actions in all applications other than itself. If you followed the instructions carefully, Silk Test Workbench captured only the actions performed on the sample application Web site. Silk Test Workbench repeats these actions during playback.

Your script should look similar to the following sample.

```
Imports SilkTest.Ntf.XBrowser
Public Module Main
    Dim _desktop As Desktop = Agent.Desktop
```

```
    Public Sub Main()
        _desktop.Control("control1").TypeKeys("9")
        With _desktop.BrowserApplication("webBrowser")
            With .BrowserWindow("browserWindow")
                .DomTextField("autoquoteZipcode").SetText("92121")
                .DomTextField("autoquoteEMail").SetText("jsmith@gmail.com")
                .DomRadioButton("autoquoteVehicle0").Select()
                .DomButton("autoquoteNext").Select()
                .DomTextField("autoquoteAge").SetText("42")
                .DomRadioButton("autoquoteGender0").Select()
                .DomRadioButton("autoquoteType1").Select()
                .DomButton("autoquoteNext").Select()
                .DomTextField("autoquoteYear").SetText("2010")
                .DomElement("img").DomClick(MouseButton.Left, New Point(8, 9))
                .DomElement("lexus").DomClick(MouseButton.Left, New Point(87,
7))
                .DomElement("img3").DomClick(MouseButton.Left, New Point(11,
10))
                .DomElement("rX400").DomClick(MouseButton.Left, New Point(96,
11))
                .DomRadioButton("autoquoteFinInfo2").Select()
                .DomButton("autoquoteNext").Select()
                .DomLink("home").Select()
            End With
        End With

    End Sub
End Module
```

Your script may not exactly match the preceding example. Different users interact with applications differently. For example, when filling out a form, some users click from field to field and others use the `Tab` key. Silk Test Workbench records these actions differently, though they achieve the same results. Your script should play back correctly regardless of these differences.

Playing Back the Recorded Script

Once you have recorded and saved your script, you can play it back to verify that the script works.

1. Perform one of the following steps:
   - Choose **Actions** > **Playback**.
   - Click **Playback** on the toolbar.

   The **Playback** dialog box opens. This dialog box lets you determine how the result is saved.
2. In the **Result description** text box, type `Initial test results for the recorded test`.
3. Click **OK**.
4. If multiple browsers that are supported for replay are installed on the machine, the **Select Browser** dialog box opens. Select the browser and click **Run**.

   Each result is identified with a unique test run number.

   Silk Test Workbench minimizes and the script plays back. During playback, the actions you performed while recording the script are played back on the screen against the sample application. When playback completes successfully, the **Playback Complete** dialog box opens.
5. Click **Go to Result**. The **Result** window opens.

Analyzing Results: Introduction

After playing back a script, Silk Test Workbench generates a test result. A test result contains information about the playback of the script. Information such as the name of the script, the run number, the date and time each step executed, the pass or fail status of each step, and other important information.

Enhancing the Script: Introduction

Enhancing a test includes making updates to the existing test to ensure that it works with newer versions of the test application. For example, to handle and verify varying conditions in the test application you can insert test logic. Additionally, to increase the readability of a test or to remind yourself or others about important aspects of the test, you can insert a message box.

These are just some of the ways in which you can use Silk Test Workbench to enhance existing tests to create more powerful, robust, and flexible tests.

Inserting a Verification

A verification is test logic that evaluates a user-defined condition, and then sends a pass/fail message to the playback result.

In this lesson, you will insert a verification to ensure that the quote uses the correct vehicle model.

1. Select the following text that defines the model type for the auto quote.
   ```
   .DomElement("RX400").DomClick(MouseButton.Left, New Point(96, 11))
   ```
2. Navigate to the page in the instant quote wizard where the **Model** type is specified and note a different model type.
   For example, **GS430** is a model type for the **Lexus** make.
3. In Silk Test Workbench, change the model type.
   Change the following textContents code from:
   ```
   .DomElement("RX400").DomClick(MouseButton.Left, New Point(96, 11))
   ```
   to:
   ```
   .DomElement("GS430").DomClick(MouseButton.Left, New Point(96, 11))
   ```
4. To compare the expected value with the actual value and add a comment, type:
   ```
   Workbench.Verify ("GS430", .DomTextField("modelCombo").Text, "The model
   type is correct")
   ```

You have successfully enhanced the recorded script by inserting test logic that verifies the value of a property in the sample application.

Creating and Storing Application Data in a Local Variable

Variables enhance tests by providing the ability to store data values for use in other parts of the script. Data can also be output to other types of files.

The sample application displays a unique email address on the **Get Instant Auto Quote** page. The text on this page containing the email address is the property value of a control on the page.

In this lesson, you will store this text to the local variable *stremailAddr*.

1. In the script, navigate to the email value.
   The code should look similar to the following:
   ```
   .DomTextField("autoquoteEMail").SetText("jsmith@gmail.com")
   ```
2. Insert the following code following the email value:
   ```
   Dim StremailAddr As String
       StremailAddr = .DomTextField("autoquoteEMail").Text
   ```
   This step creates a new local variable, *StremailAddr*, that stores the email address text from the **Get Instant Auto Quote page** page to the local variable.
3. To include output that displays the variable text during playback, include a `Console.Write` command in your script.

For example, include:

```
Console.Write (StremailAddr)
```

4. To view the console output, choose **View** > **Output**. The **Output** window opens. When you playback a script, the **Output** window is populated.

To confirm that the test captures the property value and stores it properly, play back the test and review the result.

Playing Back and Analyzing the Enhanced Script

Now that you have made several enhancements to the recorded script, play back the script and analyze the result.

1. Perform one of the following steps:
   - Choose **Actions** > **Playback**.
   - Click **Playback** on the toolbar.

   The **Playback** dialog box opens.
2. In the **Result description** text box, type `Enhanced test results for the script`.
3. Click **OK**.
4. If multiple browsers that are supported for replay are installed on the machine, the **Select Browser** dialog box opens. Select the browser and click **Run**. Silk Test Workbench plays back the enhanced script.
5. Click **Go to Result** on the **Playback Complete** dialog box.

   The **Result** window opens to the **Summary** tab by default.

   The **Summary** tab shows that the script passed, which means it played back successfully without any errors, or failed verifications.
6. Click the **Passed (1)** tab.

   The number in parentheses indicates the total number of verifications that passed. The **Test Steps** pane displays the verification step and the **Result Detail** column displays the pass text description of the verification.
7. Click the **Details** tab to display the result of every action.
8. In the **Output** window, scroll down to the line that shows the result of storing the email address to a variable.

   The text is similar to the following:

   ```
   jsmith@gmail.com
   ```
9. Click the **Passed** tab to see the results of the verification for the model type.

   The text is similar to the following:

   ```
   Main:Verify Passed: The model type is correct
   ```

   The verification results also display in the **Result** and **Result Detail** columns.

Congratulations! You have successfully created a script that reliably tests the sample application. In the next lesson, you will learn about several advanced testing concepts and features such as how to quickly and easily execute a script within another script.

Executing a Script Within a Script: Introduction

In this tutorial, you created a single script that performs every action required to receive an auto insurance quote from the web application. A single script is useful when implementing a basic test case against a simple application. However, most software testing requires a more rigorous approach that involves testing every aspect of an application. An additional requirement is the ability to rapidly update existing scripts whenever the test application changes.

To provide an efficient means for solving these testing challenges, Silk Test Workbench supports modular testing, in which you can "chunk" common sets of actions of a particular testing solution into a single test, and then reuse the script in other scripts that require the same set of actions.

Modular Testing

Before creating visual tests, scripts, and other Silk Test Workbench assets to build application testing solutions, it is a good practice to plan a testing strategy.

It is not necessary to include all the parts of a specific test solution in a single visual test or script and is not usually beneficial to do so.

Typically, the most efficient testing approach is a modular approach. Think of your application testing in terms of distinct series of transaction units.

For example, testing an online ordering system might include the following distinct transaction units:

- Log on to the online system
- Create a customer profile
- Place orders
- Log off the online system

If one test is created to handle all of these distinct units and there are ten different scenarios that use this test, you would need to record ten different tests to handle the scenarios. If any change occurs to the application, for example if an extra field is added to the logon window, ten different tests would require a change to accommodate data input to the new field.

Rather than creating one visual test or script that tests all of these transaction units, and then recreating it ten times for each scenario, it may be more beneficial to create separate tests as test "modules" that handle each one of these transaction units. If a separate test is created for each of the separate transaction units and reused for each of the test scenarios, then only the test that handles the logon transaction unit would require change.

Now that you understand the basics of modular testing, you are ready to create a second test and add it to the test you created in the previous lessons.

Recording the Second Script

In this section of the lesson, you record a second script for the tutorial and learn an alternate way to create a script.

1. Choose **File** > **New**. The **New Asset** dialog box opens.
2. Select **.NET Script** from the asset types list, and then type a name for the script in the **Asset name** text box.
   For this tutorial, type `AddAccount` for the name.
3. Check the **Begin Recording** check box to start recording immediately.
4. Click **OK** to save the script as an asset and begin recording. The **Select Application** dialog box opens.
5. Select the **Web** tab.
6. Select **Internet Explorer** from the list.
7. From the **Home** page of the sample application, click **Sign Up** in the **Login** section. The **Create A New Account** page opens.
8. Provide the following information in the appropriate fields.
   Press the `Tab` key to move from one field to the next.

Field Name	Value
First Name	Pat
Last Name	Smith

Field Name	Value
Birthday	February 12, 1990 **Note:** Click the down arrow next to the month and year in the calendar control to change the month and year and then select 12 on the calendar.
E-Mail Address	smith@test.com
Mailing Address	1212 Test Way
City	San Diego
State	CA
Postal Code	92121
Password	test

9. Click **Sign Up**.
10. Click **Continue**. The contact information is displayed.
11. Click **Home** near the top of the page to return to the home page where recording started.
12. Click **Log Out**.
13. Press `Alt+F10` to complete recording. The **Recording Complete** dialog box opens.
14. Click **Save**. The script opens in the **Code** window.

Inserting One Script Within Another

In this section of the lesson, you will learn how to insert the second script, which adds a user account, in the original script before the code that perform the request for an auto quote.

Executing scripts within scripts is a powerful method for efficiently testing the same basic actions in scripts.

Tip: When inserting a script within another script, it is important to ensure that any test applications are in the correct initial playback state.

1. Choose **File** > **Open**. The **Asset Browser** opens.
2. Select **.NET Script** in the left pane to display the list of scripts.
3. From the list, double-click `AutoQuote` to open it.

   `AutoQuote` is the first test that you created in this tutorial.
4. In the **Code** window, position the cursor after the `Public Sub Main()` code, press `Enter` to add a new line, and type:

   ```
   Workbench.RunScript ("AddAccount")
   ```

   where *`AddAccount`* is the name of the second script that you created.

   Because we want to add the account information before we execute the quote steps, we added the `Workbench.RunScript` command before the `With` statement. To execute the *AddAccount* script after the quote steps, add the `Workbench.RunScript` command after the `End With` statement.

Responding to Playback Errors: Introduction

Errors encountered during playback can be caused by a variety of factors, such as changes in the test application and improper workflow. Quickly diagnosing and fixing these errors minimizes test maintenance and allows for a more efficient team testing effort.

First, begin this lesson by playing back the modular script you created in the previous lesson.

Playing Back the Modular Script

In the previous lesson, you created a modular script by inserting AddAccount into the AutoQuote script.

In this section of the lesson, you will play back the modular script and encounter an error during playback.

1. With the AutoQuote script open, click **Playback** on the toolbar. The **Playback** dialog box opens.
2. In the **Result description** text box, type `Responding to errors in a modular test`.
3. Click **OK**.
4. If multiple browsers that are supported for replay are installed on the machine, the **Select Browser** dialog box opens. Select the browser and click **Run**.

   During playback, the test stops on the **Create A New Account** page and an error message opens.

   This error occurs because the database requires a unique email address for each customer record. Since you have already entered the email address during the recording of the AddAccount script, the email address already exists in the database and the test fails.

Reviewing the Result

Review the results of the script.

1. Click **End** to stop playback. The **Playback Complete** dialog box opens.
2. Click **Go to Result**. The AutoQuote result appears with the **Summary** tab displayed by default.

   The **Summary** tab displays the overall details of the test run. Note that the **Visual tests or .NET Scripts (number of times each ran)** field lists AutoQuote(1) and the inserted script, AddAccount(1).
3. Click the **Details** tab.
4. Scroll down to the steps in blue text.

   By reviewing the **Result** and **Result Detail** columns, you can quickly find information about any errors that occurred during playback.

   Note: The **Failed** tab does not display steps containing playback errors. It only displays failed verifications.

You have learned how to diagnose playback errors. For the purposes of this tutorial and to successfully replay the script once, you can manually modify the email in the script to avoid the error. To do so, change

```
smith@test.com
```

in the *AddAccount* script to

```
psmith@test.com
```

Help on Help

This section includes information about:

- Silk Test Workbench Help
- Typographic Conventions Used in the Help

Silk Test Workbench Help

The Silk Test Workbench Help includes conceptual overviews and procedural how-to topics. These topic types allow you to navigate from general to more specific information as needed.

Conceptual Overviews	The conceptual overviews provide information about product architecture, components, and other information you need to work with Silk Test. At the end of most of the overviews, you will find links to related, more detailed information.
How-To Procedures	The how-to procedures provide step-by-step instructions. At the end of most of the procedures, you will find links to related procedures. Additionally, most of the conceptual overviews provide links to the pertinent procedures.

Typographic Conventions Used in the Help

The following typographic conventions are used in the Silk Test Workbench Help.

Convention	Used to indicate
`Monospace type`	Source code and text that you must type.
Boldface	References to dialog boxes and other user interface elements.
Italics	Identifiers, such as variables. Italicized text is also used to emphasize new terms.

Ways to Get Help

The Silk Test Workbench **Help** menu contains the following commands:

- Click **Contents** to view the table of contents for the Silk Test Workbench Help.
- Click **Search** to search for topics by using a full-text search.
- Click **Index** to search for topics by using a keyword index.
- Click **Tutorial** to run the Silk Test Workbench tutorial, which lets you get familiar with Silk Test Workbench by creating and running visual tests with a sample application.
- Click **About Silk Test Workbench** to display the currently installed version, copyright, and system information.

You can also access the Silk Test Workbench Help by clicking the **Help** button located on selected dialog boxes.

Printing a Help Topic

1. Click the **Contents** tab.

   If the **Contents** tab is not visible, click **Show** to display the navigation pane.
2. Select the topic to print.
3. Click **Print**.
4. To print multiple topics, select a book, click **Print**, and then click the **Print the selected heading and all subtopics** option button.

   The topics print continuously, rather than printing each heading on a separate page.

Enabling or Disabling Usage Data Collection

To help Micro Focus improve your overall testing experience, Micro Focus would like to collect some information on how you use Micro Focus software and services. By accepting the terms of the License Agreement while installing Silk Test Workbench, you allow Micro Focus to collect information about how you use Silk Test Workbench and about the computer system on which Silk Test Workbench is installed. Micro Focus does not collect any personally identifiable information, for example your name or address, or any of your data files, for example scripts or passwords. By allowing Micro Focus to collect this information, you assist Micro Focus in identifying trends and usage patters.

To enable or disable the collection of usage data by Micro Focus:

1. Click **Help** > **About Silk Test Workbench** in the menu.
2. In the About dialog box, click **Customer Feedback Options**.
3. Select one of the following options:

- To enable usage data collection, click **Yes, I am willing to participate**.
- To disable usage data collection, click **No, I would not like to participate**.

4. Click **OK**.

Contacting Micro Focus

Micro Focus is committed to providing world-class technical support and consulting services. Micro Focus provides worldwide support, delivering timely, reliable service to ensure every customer's business success.

All customers who are under a maintenance and support contract, as well as prospective customers who are evaluating products, are eligible for customer support. Our highly trained staff respond to your requests as quickly and professionally as possible.

Visit *http://supportline.microfocus.com/assistedservices.asp* to communicate directly with Micro Focus SupportLine to resolve your issues, or email supportline@microfocus.com.

Visit Micro Focus SupportLine at *http://supportline.microfocus.com* for up-to-date support news and access to other support information. First time users may be required to register to the site.

Information Needed by Micro Focus SupportLine

When contacting Micro Focus SupportLine, please include the following information if possible. The more information you can give, the better Micro Focus SupportLine can help you.

- The name and version number of all products that you think might be causing an issue.
- Your computer make and model.
- System information such as operating system name and version, processors, and memory details.
- Any detailed description of the issue, including steps to reproduce the issue.
- Exact wording of any error messages involved.
- Your serial number.

To find out these numbers, look in the subject line and body of your Electronic Product Delivery Notice email that you received from Micro Focus.

Creating Tests

Describes the process and considerations for creating visual tests and scripts.

Creating Visual Tests

Visual tests can be a series of simple recorded actions, or they can be complex, involving function calls or ActiveData.

Recording is the best way to quickly create visual tests. Using the Record feature enables you to become familiar with how Silk Test Workbench generates test steps for a visual test. This provides a basic understanding of visual test creation, which gives users insight into creating efficient test solutions.

During recording, all actions except interaction with Silk Test Workbench are recorded. However, it is possible to interrupt recording to insert a verification. You can modify a visual test after initial recording is finished.

Recording a Visual Test

During recording, Silk Test Workbench records all interactions in the test application (except interaction with Silk Test Workbench itself) until recording is stopped. After you have finished recording, you can modify the visual test you have generated to add and remove steps.

1. *Optional:* To be able to update recorded controls from the screen preview, when testing against a web application, enable the capturing of browser controls before recording a visual test.

   For additional information, see *Setting Record Output Options*.

   Note: Enabling control capture during the recording of web applications might decrease the performance of your visual tests. Micro Focus recommends disabling control captures.
2. Choose **File** > **New**. The **New Asset** dialog box opens.
3. Select **Visual test** from the asset types list, and then type a name for the visual test asset in the **Asset name** text box.

   Note: The visual test asset is created in the active project by default, but you can create it in another project by selecting it from the **Project** list.
4. *Optional*: Click **Options** if you want to change Silk Test Workbench record options.
5. Check the **Begin Recording** check box to start recording immediately.
6. Click **OK** to save the visual test as an asset and begin recording.

   If you click **OK** without checking the **Begin Recording** check box, Silk Test Workbench saves the visual test as an asset and displays the **Visual Navigator**.

   The **Select Application** dialog box opens.

   Note: If the application that you want to test does not appear in the list, uncheck the **Hide processes without caption** check box. This option, checked by default, is used to filter only those applications that have captions.
7. If you have not set an application configuration for the current project, select the tab that corresponds to the type of application that you are testing:

   - If you are testing a standard application that does not run in a browser, select the **Windows** tab.
   - If you are testing a web application or a mobile web application, select the **Web** tab.
   - If you are testing a native mobile application, select the **Mobile** tab.

8. To test a standard application, select the application from the list.
9. To test a web application or a mobile web application, select one of the installed browsers or mobile browsers from the list.
   a) Specify the web page to open in the **Enter URL to navigate** text box. If an instance of the selected browser is already running, you can click **Use URL from running browser** to record against the URL currently displayed in the running browser instance.
   b) *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
   For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.
   c) *Optional:* Select an **Orientation** for the browser window.
   d) *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.
10. To test a native mobile application (app):
   a) Select the mobile device, on which you want to test the app, from the list.
   b) Click **Browse** to select the app file or enter the full path to the app file into the **Mobile app file** text field.
   Silk Test Workbench supports HTTP and UNC formats for the path.
   Silk Test Workbench installs the app on the mobile device or emulator.
11. Click **OK**. Silk Test Workbench minimizes and the application and the **Recording** window opens.
12. Record the actions that you want to test.
   During recording the Silk Test Workbench icon on the task bar flashes. For information about the actions available during recording, see *Actions Available During Recording*.
13. If you are recording against a web application on one of the supported browsers, and you want to interact with a control that you cannot access directly in the UI, for example because the control is hidden by another control, click **Toggle Hierarchy View** in the **Recording** window and select the control from the control hierarchy tree.
   This functionality is available if you are testing against one of the following browsers:
   - Microsoft Edge.
   - Apple Safari.
   - Mozilla Firefox 41 or later.
   - Google Chrome 50 or later.
   - A mobile browser.
14. *Optional:* Add verification logic to the test.
   - If you are testing a standard or web application, click `Ctrl+Alt` to add verification logic to the test. Silk Test Workbench temporarily suspends recording and displays the **Test Logic Designer** wizard.
     Follow the wizard through the process and click **Finish** to close the wizard and continue recording.
   - If you are testing a web application on Microsoft Edge or a mobile application, click on the object that you want to verify and click **Add Verification** in the **Choose Action** dialog box.

   For additional information about adding a verification, see *Adding a Verification to a Script while Recording*.
15. Stop recording by pressing `Alt+F10`, clicking **Stop** in the **Recording** window, or clicking the Silk Test Workbench taskbar icon. The **Recording Complete** dialog box opens. If the **Do not show this message again** check box is checked in the **Recording Complete** dialog box, this dialog box does not appear after recording is stopped. In this case, the visual test displays in the **Test Steps** pane.
16. Perform one of the following steps:
   - Click **Playback** to close the **Recording Complete** dialog box and save the test script.

The **Playback** dialog box appears, which allows you to specify result information and playback the test.

- Click **Go To Visual test** to open the visual test in the **Test Steps** pane of the **Visual Navigator**.
- Click **Save** to save the visual test and exit the **Recording Complete** dialog box.

  The visual test appears in the **Test Steps** pane of the **Visual Navigator**.

When the recorded test displays in the **Visual Navigator**, you can play it back at any time.

Recording a Visual Test From the Start Screen

During recording, everything you do, except your interaction with Silk Test Workbench, is recorded until you stop the session. After you have finished recording, you can modify the visual test to add and remove steps.

1. *Optional:* To be able to update recorded controls from the screen preview, when testing against a web application, enable the capturing of browser controls before recording a visual test.

   For additional information, see *Setting Record Output Options*.

   Note: Enabling control capture during the recording of web applications might decrease the performance of your visual tests. Micro Focus recommends disabling control captures.
2. Click **Record New Visual Test** on the **Start Screen**. The **Select Application** dialog box opens.
3. If you have not set an application configuration for the current project, select the tab that corresponds to the type of application that you are testing:
   - If you are testing a standard application that does not run in a browser, select the **Windows** tab.
   - If you are testing a web application or a mobile web application, select the **Web** tab.
   - If you are testing a native mobile application, select the **Mobile** tab.
4. To test a standard application, if you have not set an application configuration for the current project, select the application from the list.
5. To test a web application or a mobile web application, select one of the installed browsers or mobile browsers from the list.
   a) Specify the web page to open in the **Enter URL to navigate** text box. If an instance of the selected browser is already running, you can click **Use URL from running browser** to record against the URL currently displayed in the running browser instance.
   b) *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
   For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.
   c) *Optional:* Select an **Orientation** for the browser window.
   d) *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.
6. To test a native mobile application (app):
   a) Select the mobile device, on which you want to test the app, from the list.
   b) Click **Browse** to select the app file or enter the full path to the app file into the **Mobile app file** text field.

      Silk Test Workbench supports HTTP and UNC formats for the path.

      Silk Test Workbench installs the app on the mobile device or emulator.
7. Click **OK**. Silk Test Workbench minimizes and the application and the **Recording** window opens.
8. Record the actions that you want to test.

   During recording the Silk Test Workbench icon on the task bar flashes. For information about the actions available during recording, see *Actions Available During Recording*.
9. If you are recording against a web application on one of the supported browsers, and you want to interact with a control that you cannot access directly in the UI, for example because the control is

hidden by another control, click **Toggle Hierarchy View** in the **Recording** window and select the control from the control hierarchy tree.

This functionality is available if you are testing against one of the following browsers:

- Microsoft Edge.
- Apple Safari.
- Mozilla Firefox 41 or later.
- Google Chrome 50 or later.
- A mobile browser.

10. Stop recording by pressing `Alt+F10`, clicking **Stop** in the **Recording** window, or clicking the Silk Test Workbench taskbar icon. The **Recording Complete** dialog box opens. If the **Do not show this message again** check box is checked in the **Recording Complete** dialog box, this dialog box does not appear after recording is stopped. In this case, the visual test displays in the **Test Steps** pane.
11. Perform one of the following steps:

    - Click **Playback** to close the **Recording Complete** dialog box and save the test script.

      The **Playback** dialog box appears, which allows you to specify result information and playback the test.
    - Click **Go To Visual test** to open the visual test in the **Test Steps** pane of the **Visual Navigator**.
    - Click **Save** to save the visual test and exit the **Recording Complete** dialog box.

      The visual test appears in the **Test Steps** pane of the **Visual Navigator**.

    When the recorded test displays in the **Visual Navigator**, you can play it back at any time.

Recording a Visual Test that Tests Multiple Test Applications

You can record a test that tests multiple test applications. For example, if you are testing an application that modifies a database and you use a database viewer tool to verify the database contents, you must add an additional application configuration for the database viewer tool.

1. Record a visual test or create one manually for the primary application that you want to test.
2. Open the visual test.
3. Click the **<<Start>>** step in the **Task** pane.
4. In the **Properties** pane, navigate to the **Application Configurations** category.
5. Click into the **Application Configurations** text box.
6. Click **Add Application Configuration**. The **Select Application** dialog box appears.
7. Select the tab that corresponds to the type of application that you want to additionally test.

    - If you are testing a standard application that does not run in a browser, select the **Windows** tab.
    - If you are testing a web application or a mobile web application, select the **Web** tab.
    - If you are testing a native mobile application, select the **Mobile** tab.
8. To test a standard application, select the application from the list.
9. To test a web application or a mobile web application, select one of the installed browsers or mobile browsers from the list.
    a) Specify the web page to open in the **Enter URL to navigate** text box. If an instance of the selected browser is already running, you can click **Use URL from running browser** to record against the URL currently displayed in the running browser instance.
    b) *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
    For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.

c) *Optional:* Select an **Orientation** for the browser window.
d) *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.

10. To test a native mobile application (app):
a) Select the mobile device, on which you want to test the app, from the list.
b) Click **Browse** to select the app file or enter the full path to the app file into the **Mobile app file** text field.

Silk Test Workbench supports HTTP and UNC formats for the path.

Silk Test Workbench installs the app on the mobile device or emulator.

11. Click **OK**. Silk Test Workbench adds a new application configuration to the **Properties** pane.

12. Record additional steps for the visual test using the new application configuration.

Navigating from a Locator to an Object Map Entry in a Visual Test

If you want to see more than the **ID** of an object map entry, you can easily see the raw locator that will be used by the Open Agent when the command is executed by doing the following:

1. Open a visual test.
2. Select a step in the **Test Steps** pane. In the **Properties** pane, the **Locator** field displays the locator text.
3. Place your cursor in the **Locator** field text. The **Open Object Map** button appears to the right.
4. Click the **Open Object Map** button. The **Object Map** window opens with the proper item selected in the tree view.

Configuring a Visual Test to Launch an Application that Uses the Java Network Launching Protocol (JNLP)

Applications that start using the Java Network Launching Protocol (JNLP) require additional configuration in Silk Test Workbench. Because these applications are started from the Web, you must manually configure the application configuration to start the actual application as well as the application that launches the "Web Start". Otherwise, the visual test will fail on playback unless the application is already running.

1. Record a visual test for the application that you want to test.
2. Click the **<<Start>>** step in the **Task** pane.
3. In the **Properties** pane, navigate to the **Application Configurations** category.
4. Click the **Application Configurations** field and then click **Add Application Configuration**. The **Select Application** dialog box appears.
5. Select the **Windows** tab.
6. Select the Java application that you want to test.
7. Click **OK**. Silk Test Workbench adds the new application configuration.
8. Modify the first application configuration, **Config 1**, to use the `javaws.exe` to download and launch the Web Start.
   a) Click into the **Config 1** field and click **Edit Application Configuration**.
   b) In the **Executable** field, type the executable name and file path of the Web Start.
   For example, you might type `%ProgramFiles%\Java\jre6\bin\javaws.exe`.
   c) In the **Working directory** field, type a temporary directory location for the file that you will download.
   For example, you might type `C:\temp` where *temp* is a directory on your local C drive.

d) In the **Locator** field, specify the locator that identifies the main window of the application that you are testing.

   In most cases, you will not have to change this because when you recorded the visual test, the correct locator was recorded.

   For example, if you are testing the SwingSet3 JNLP application, the locator is `SwingSet3`.

e) In the **Command Line Arguments** field, type the JNLP URL that starts the test application.

   For instance, to use the SwingSet3 JNLP application, type `http://download.java.net/javadesktop/swingset3/SwingSet3.jnlp`.

f) Delete any value in the **Command Line Pattern** field.

g) To change the number of milliseconds that Silk Test Workbench attempts to find the locator that is specified for the base state before timing out, type a value into the **Timeout** field.

   By default **Timeout (milliseconds)** is set to 30000, which is sufficient for most applications.

h) Leave the **Execute base state** text box set to **True**

9. Modify the new application configuration to use the Java application that you want to test.

a) In the **Executable** field, type the executable name and file path of the application that you want to test.
   For example, to use the SwingSet3 JNLP application, type `*\javaw.exe`.

b) Leave the **Working directory** field empty.

c) In the **Locator** field, specify the locator that identifies the main window of the application that you want to test.

   In most cases, this matches the name of the locator that you specified in **Config 1**.

   For example, if you are testing the SwingSet3 JNLP application, the locator is `SwingSet3`.

d) Delete any value in the **Command Line Arguments** field.

e) In the **Command Line Pattern** field, type the command line arguments pattern for the test application.

   An application configuration that contains a command line arguments pattern enables only processes for testing that match both the process name and the command line arguments pattern. If no command line arguments pattern is defined, all processes with the specified process name executable pattern are enabled.

   For instance, to use the SwingSet3 JNLP application, type `*SwingSet*`. This argument ensures that only the SwingSet3 Java application is launched.

f) To change the number of milliseconds that Silk Test Workbench attempts to find the locator that is specified for the base state before timing out, type a value in the **Timeout** text box.

   By default **Timeout (milliseconds)** is set to 30000, which is sufficient for most applications.

g) Check the **Execute Base State** check box.

When you playback the visual test, the JNLP application starts as expected.

Configuring a Visual Test to Launch in a New Browser Window

At times you may want to launch a separate browser window for a visual test rather than using the existing browser window. For instance, when you execute a Silk Test Workbench visual test from Silk Central Test Manager, the test uses the existing Silk Central window. To return to Silk Central, the user must click **Back** in the browser window. To avoid this behavior you can configure your test to launch a separate browser window for tests that you will execute from Silk Central.

1. Record a visual test for the application that you want to test.

2. In the **Test Steps** pane, click the `<<Start>>` step.

3. In the **Properties** pane, navigate to the **Application Configurations** category.

4. Click the **Application Configurations** field and then click **Add Application Configuration**. The **Select Application** dialog box appears.
5. Click into the application configuration and click **Edit Application Configuration**. The **Edit Application Configuration** dialog box appears.
6. From the **Browser Type** list, select the browser to use.
7. Uncheck the **Execute Base State** check box.
8. In the **Test Steps** pane, choose **Insert** > **External Program**. Silk Test Workbench adds `Run program " (Show normal)` to the **Test Steps** pane.
9. In the **Properties** pane, navigate to the **Command** category.
10. Click into the **Path name** field and click **Select file**. The **Select file** dialog box appears.
11. Navigate to the executable file for the browser, and then click **Open**.

    For example, to use Internet Explorer, specify `C:\Program Files\Internet Explorer \iexplore.exe`.
12. Verify that the **Operation** list box specifies **Open**.
13. Type the URL of the web application that you want to test into the **Parameters** field.

    For instance, to test the sample web application, type `http://demo.borland.com/InsuranceWebExtJS/index.jsf`.

Viewing Specific Steps in a Visual Test

You can customize which test steps display in a visual test to only view certain steps. Filtering steps to display in a visual test help to quickly understand the nature of a test, and focus on only the steps that require review. If recording additional actions in a visual test that only displays certain types of steps, only the recorded actions that meet the display criteria display after recording. Change the view option to **Steps and Screens** to display the other newly recorded steps.

1. With the visual test open in the **Test Steps** pane, choose **Actions** > **View**.
2. Select a view option for steps in the visual test:
3. Click **Steps Only** to hide any steps in the visual test only interact with screens.
4. Click **Screens Only** to hide any steps that perform test actions.
5. Click **Steps and Screens** to view all test steps in a visual test.

   This is the default view setting.
6. Click **Step Description** to display the **Description** column in the **Test Steps** pane. This column is hidden by default. Steps that include a description show the **Test Step Description** icon in the **Description** column. Enter a description for a test step by updating the **Description** value for the step in the **Properties** pane.

Identifying a Control for a Visual Test in the Test Application

When you record a test, Silk Test Workbench automatically identifies controls. When the visual test is recorded, you can identify visible controls in the application under test to change controls that are used in the visual test or to manually insert test steps into the visual test.

1. In the **Test Steps** pane, select the step for which you want to change the control.
2. In the **Properties** pane, expand the **Identify a control** menu in the **Locator** text box.
3. Select **Application Under Test**.
4. Move the cursor to the application that you are testing.

   Controls appear highlighted with a dark border as the cursor passes over them in the application.

Tip: Press Esc to exit from Identify mode and return to the visual test.

5. Click the control that you want to use as it is highlighted.

   Note: If editing an automation step that accesses a different class, Silk Test Workbench issues a message indicating that the class accessed by the step has changed and asks if you want to apply the change. Click **Yes** to update the **Locator** with the new class and control information, or click **No** to return to the **Visual Navigator** without identifying a control.

   The **Locator** text box is updated with the newly selected control information and this control is used during playback.

Identifying a Control for a Visual Test in Screen Preview

When you record a test, Silk Test Workbench automatically identifies controls. When the visual test is recorded and the application under test is no longer available, you can use the **Screen Preview** to change controls that are used in the visual test or to manually insert test steps into the visual test.

Note: To identify a control name in the **Screen Preview** of a visual test against a web application, the capturing of browser controls needs to be enabled during the recording of the visual test. For additional information, see *Setting Record Output Options*.

1. In the **Test Steps** pane, select the step for which you want to change the control.
2. In the **Properties** pane, expand the **Identify a control** menu in the **Locator** text box.
3. Select **Screen Preview**.
4. Move the cursor to the captured screen in the **Screen Preview**.

   Tip: Press Esc to exit from Identify mode in the **Screen Preview** and return to the visual test.

   Controls appear highlighted with a dark border as the cursor passes over them in the **Screen Preview**.
5. Click the control that you want to use as it is highlighted.

   Note: If editing an automation step that accesses a different class, Silk Test Workbench issues a message indicating that the class accessed by the step has changed and asks if you want to apply the change. Click **Yes** to update the **Locator** with the new class and control information, or click **No** to return to the **Visual Navigator** without identifying a control.

   The **Locator** text box is updated with the newly selected control information and this control is used during playback.

Identifying a Control for a Visual Test with the Identify Object Dialog

When you record a visual test, Silk Test Workbench automatically identifies controls. When the visual test is recorded, you can change the controls that are used in the visual test or manually insert additional test steps into the visual test. Use the **Identify Object** dialog to identify non-visible controls, for example containers, in the application under test.

1. In the **Test Steps** pane, select the step for which you want to change the control.
2. In the **Properties** pane, expand the **Identify a control** menu in the **Locator** text box.
3. Select **Identify Object Dialog**.
4. If you are testing a web application, the **Select Browser** dialog box opens. Select the browser on which you want to identify the control and click **Start Identify**. The **Identify Object** dialog box opens.
5. Click **Start Identify**.

6. Move the cursor to the application that you are testing. Controls appear highlighted as the cursor passes over them in the application. The related locator string or object map item shows in the **Selected Locator** text box.
7. Click the control that you want to use as it is highlighted.
8. *Optional:* Click **Show Locator Details** to display the locator tree and any related attributes in the **Locator Attribute** table.
9. *Optional:* You can replace a recorded locator attribute with another locator attribute from the **Locator Attribute** table.

   For example, your recorded locator might look like the following:

   ```
   /BrowserApplication//BrowserWindow//input[@id='loginButton']
   ```

   If you have a `textContents Login` listed in the **Locator Attribute** table, you can manually change the locator to the following:

   ```
   /BrowserApplication//BrowserWindow//input[@textContents='Login']
   ```

   The new locator displays in the **Selected Locator** text box.
10. Click **Test** to verify that the locator recognizes the correct control.
11. To add a new step that uses the locator to the visual test, click **Paste**. The **Select Action** dialog box opens. Select the action that should be performed against the control and click **OK**.

Recording Additional Actions at a Specific Test Step

Once a visual test step is created, you can open it and record additional actions from any point in the test. This allows you to update an existing test with additional actions when needed.

1. Open an existing visual test.
2. Select the test step after which you want to record additional actions.

   Note: Recorded actions are inserted after the selected location. The application under test (AUT) does not return to the base state. Instead, the AUT opens to the scope in which the preceding actions in the test script were recorded.
3. Click **Actions** > **Record**.

   Silk Test Workbench minimizes and the **Recording** window opens.
4. Record the additional actions that you want to perform against the AUT.

   During recording the Silk Test Workbench icon on the task bar flashes.For information about the actions available during recording, see *Actions Available During Recording*.
5. To stop recording, click **Stop** in the **Recording** window.

   You can also stop recording by pressing the stop recording key combination, which by default is **Alt +F10**, or by clicking the Silk Test Workbench task bar icon.

Inserting a Screen from the Test Application into a Visual Test

Once a visual test step is created, you can open it and insert a screen capture and manually add the related test steps rather than recording additional screens and actions. Inserting a screen capture enables you to manually build a visual test that contains updated screens from the test application, which, in some cases, might be more efficient than recording the new screen.

1. Open the application that you are testing and navigate to the page or dialog that you want to capture.
2. Open an existing visual test.
3. Select the test step after which you want to insert the new screen.

Note: To view test steps and screens in the **Test Steps** pane, click **Actions** > **View** > **Steps and Screens**.

4. Choose **Insert** > **Using**.

   Silk Test Workbench minimizes and the **Recording** dialog box opens.
5. Click the screen that you want to capture, or press the hot key keystroke combination to capture the screen.

   The record locator settings option determines whether a click or hot key keystroke combination captures the screen. To change this setting, choose **Tools** > **Options** and then expand **Record** in the **Options** menu tree and and click **Locator**.

   The new `Using` step appears after the selected test step in the **Test Steps** pane.

Add the related automation test steps as necessary.

Recording an Object Map Item or a Locator Manually For a Visual Test

Before you begin, ensure that the application that you want to test is running.

Manually capture an object map item or a locator using the **Identify Object** dialog box and create a corresponding test step in a visual test.

1. Open the visual test that you want to modify. A step that corresponds with each recorded action displays in the **Test Steps** pane.
2. Click the test step in which you want to insert an object and action.

   The new object and action will be inserted below the selected test step.
3. Choose **Tools** > **Identify Object**. The **Identify Object** dialog box opens.
4. Specify the **Selection mode**.

   - **Click** – Click the object to identify the locator.
   - **Hot Key** – Specify this mode to capture the object using the keystroke combination specified in the **Keystroke** list box. Typically, you choose this mode to capture objects, such as a menu or combobox, that only appear when clicked by the user. With this mode, you select the object and then press the hot key keystroke combination to capture the locator without dismissing the object.
5. Click **Start Identify**.

   If you want to capture an object map item or a locator in Google Chrome, Silk Test Workbench detects if the selected instance of Google Chrome is started with the appropriate automation parameters, and if not, Silk Test Workbench closes Google Chrome and restarts it with the automation parameters set.
6. *Optional:* To bring the application under test into the appropriate state before recording a locator, click **Stop Identify**. The actions that you perform in the application under test are no longer recorded. To continue with the recording of a locator, click **Start Identify**.
7. Position the mouse over the object that you want to record and perform one of the following steps:

   - If you use **Click** mode, click the object that you want to identify.
   - Press the keystroke combination to capture the object with the **Hot Key** mode.

     By default, the keystroke combination is `Ctrl+Shift`.

   Silk Test Workbench lists the related locator string in the **Selected Locator** text box.
8. To refine how locators display in the **Locator Details** table, perform any of the following actions:

   - **Hide Locator Details** – To hide the **Locator Details** table, click this link.
   - **Show object map names** – Check this check box to display object map item names in the **Locator** column. Object map item names associate a logical name (an alias) with a control or a window, rather than the control or window's locator. By default, object map item names are displayed. To use locators, uncheck this check box.

- **Show full locators** – Check this check box to display the full locator name. To show only the attribute associated with the object, uncheck this check box.
- **Show properties** – Check this check box to display any attributes and attribute values for the object selected in the **Locator Details** table. You can select attributes in this table to use in the locator identification. To hide the Properties subtree and display only locator details, uncheck this check box.

9. To test that the object that displays in the **Selected Locator** text box is the object that you want to use, click **Test**. Silk Test Workbench highlights the object that corresponds with the locator in the application that you want to test.
10. To replace the locator that you recorded, select the locator that you want to use in the **Locator Details** table. The new locator displays in the **Selected Locator** text box.
11. To copy the object and create a test step in the visual step, click **Paste**. The **Select Action** dialog box opens.
12. Select the action that you want the object to perform and then click **OK**. Silk Test Workbench adds the object and the action in a new test step to the visual test.

Opening an Existing Visual Test

You can open an existing visual test from the **Start Screen** or the **Asset Browser**.

1. To open a visual test from the **Start Screen**, in the **Scripts** pane of the **Start Screen**, perform one of the following steps:
   - In the **Recent** list, click **Open Visual test '*Project name* - *Test name*'** for the appropriate visual test.
   - Click **Open** to open the **Asset Browser** and then right-click the appropriate visual test and choose **Open**.
2. To open a visual test from the **Asset Browser**, perform the following steps:
   a) Choose **View** > **Asset Browser**.
   b) Select **Visual test** in the **Asset Types** pane, right-click the appropriate visual test and choose **Open**.

   The visual test opens in the **Test Steps** pane of the **Visual Navigator**.

Saving a Visual Test

You can explicitly save a new or changed visual test. When you exit an updated visual test, you are prompted to save any changes.

Tip: You cannot undo saved changes.

1. Choose **File** > **Save As**. The **Save As** dialog box opens.
2. *Optional*: Change the visual test name and enter a description.
3. Click **OK**.

Visual Tests Overview

Visual tests comprise the basic building blocks of an automated testing solution. Silk Test Workbench uses visual tests to mimic the actions that are performed while testing an application. It controls an application or Web page in the same way that a user would by using keystrokes and mouse actions to select menus, list items, and buttons. When a visual test is recorded, Silk Test Workbench generates all the keystrokes and mouse clicks.

The actions taken to test an application, such as making menu selections, typing data, checking the way the data is processed, and so on, are represented as distinct steps in a visual test. Each step is a separate line in a visual test. These steps can be modified and played back once they are recorded.

Silk Test Workbench lets you quickly record and playback visual tests. Modify visual tests to include statements that cannot be recorded, to make amendments to reflect changes in the test application, or to create new visual tests by cutting and pasting steps from existing visual tests.

You can record visual tests automatically, manually edit visual tests, or use a combination of automatic and manual tests.

Silk Test Workbench VB .NET scripts contain a series of code lines while visual tests contain a series of test steps. Each line of code or test step executes an individual action against a test application.

Advantages of Visual Tests

Visual tests provide the following advantages:

- Record and edit tests against test applications without generating a programming language based script.
- Visually set actions and values against controls in test applications at the test step level.
- See the relationship between a text-based action step in a visual test and the object of the action.
- Take greater ownership in the testing process (for users familiar with business processes being automated).
- Leverage existing scripts and all other existing test assets by using them in visual tests.
- Recapture test applications and controls during testing without requiring access to the application being tested.
- Update screen captures automatically as test applications are updated without having to rewrite tests.
- Collaborate test project issues, assignments and other test team information using flags.
- Test against an application by playing back a visual test. Visual tests can by played back for application testing at any time. Visual tests can:
    - Run automatically at a specified time of day.
    - Run entirely unattended.
    - Interact with users to receive confidential information such as IDs and passwords.

Benefits of Visual Tests Over Scripts

Like scripts, visual tests record the actions that you use to test an application. These actions include making menu selections, typing data, clicking icons, and verifying results. Visual tests are the central testing unit in Silk Test Workbench and have the following advantages over using only scripts for a testing solution:

- Visual tests display as a series of steps in clear, non-technical language that can be easily understood by any tester.
- Visual tests are not bound to a scripting language, so any tester can create detailed, robust testing solutions without programming language knowledge.
- Visual tests are also graphically represented as thumbnail images in the Storyboard, which provides a quick overview of a visual test's flow.
- After recording a visual test, testers can use the **Screen Preview** to make updates to the visual test without requiring access to the test application.
- Visual tests facilitate the test solution building process; one tester records the test and passes it on to another tester who can make the visual test more intelligent by adding logic, validations, and loops, as well as the ability to handle unexpected situations in a specific manner.
- Visual tests do not have to be rewritten if the test application changes. Test steps can be quickly updated.
- New procedures can easily be built by copying and modifying existing visual tests.
- Visual tests facilitate a modular approach to testing and are highly re-useable. Visual tests can call other visual tests.
- Silk Test Workbench lets you collaborate and communicate test project information to other testers.

Sample Visual Tests

The following sample visual tests illustrate how to use common visual test functionality.

Sample Visual Test: Using ActiveData

This visual test sample demonstrates using ActiveData in a visual test. This visual test uses an existing ActiveData file to test a Windows test application by entering multiple names (first and last name) in a **ListView** control. Each record in the file contains a first and last name. The sample visual test cycles through the data in the file to test the windows application by entering the data in the application's **ListView** control.

Sample Visual Test: Using ActiveData - Part 1 Scenario

Test Scenario

The tester wants to use customer first and last name data stored in a spreadsheet to enter multiple name records in a Windows test application. The data exists in the spreadsheet called `Names1.xls` at the time of testing.

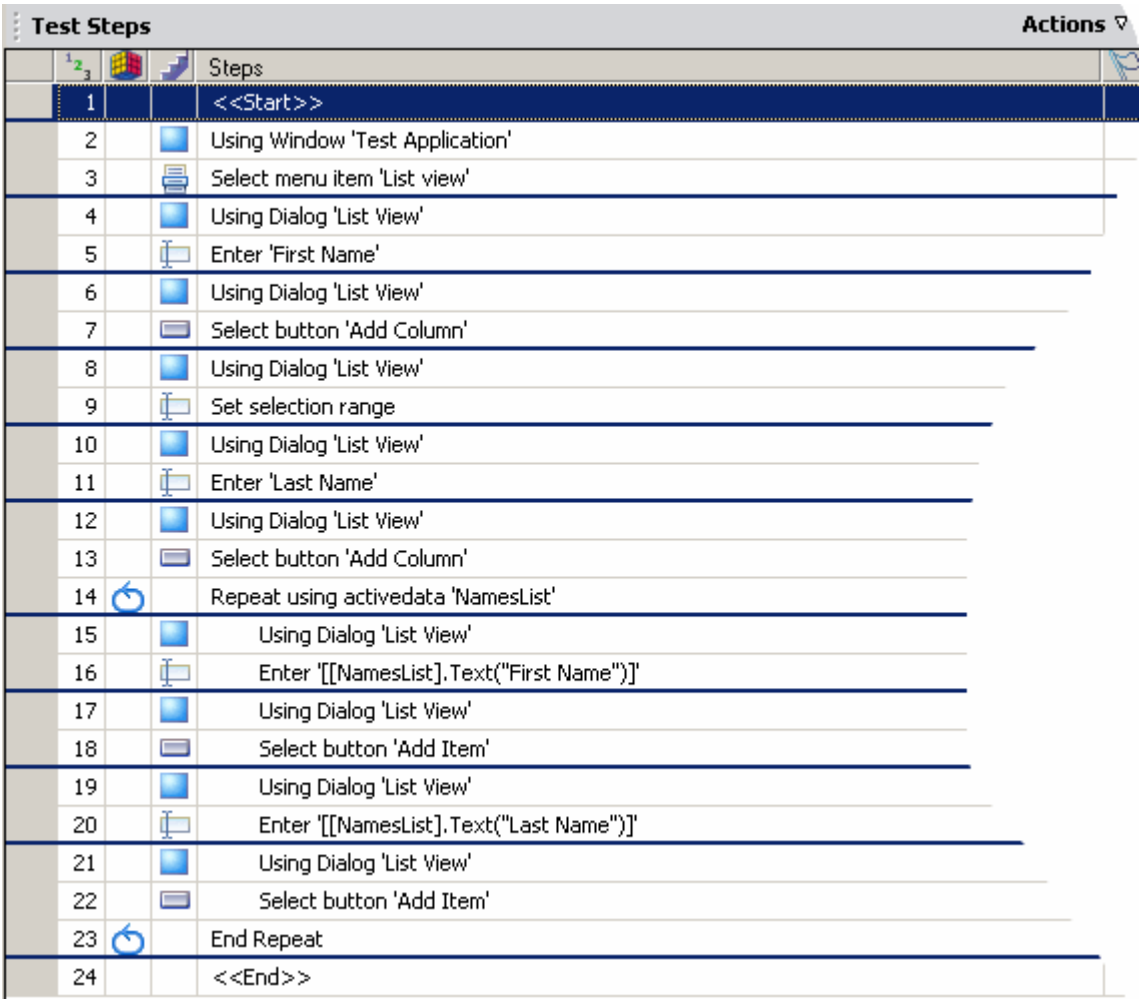
To do this, the tester does the following:

- Records accessing the **ListView** control in the Windows test application and creates a First Name and Last Name column in the control, then enters one first name/last name pair.
- Creates an ActiveData asset to be associated with the visual test. When creating the ActiveData asset, the tester selects the `Names1.xls` data file that contains the data to be used.
- Associates that ActiveData file with the visual test.
- Creates Repetition logic to repeat recorded steps that enter the first name and last name into **ListView** control of the test application.
- Modifies the properties of steps that enter the first name and last name to use data in the ActiveData file.

Sample Visual Test: Using ActiveData - Part 2 Details

Sample Visual Test

The following shows the sample visual test steps in the **Test Steps** pane based on the test scenario and highlights the progression of steps in the test.



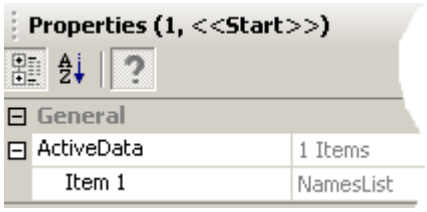
Step Details

Details of the visual test by steps is as follows:

Step 1: The `<<Start>>` step. Properties for this step display the name of the ActiveData asset used in the visual test.

In this test scenario, the **Step 1 Item 1** property in the ActiveData category shows the ActiveData asset association. This ActiveData asset contains the file `Names1.xls`. The visual test uses data in this file.

The graphic below shows the **Item 1** property of the `<<Start>>` step for this visual test.



Steps 2-3: Recorded steps that access the **ListView** by selecting the **List view** menu item in the test application. This represents the starting point for the repeated steps that follow.

Steps 4-13: Recorded steps that create First Name and Last Name columns in the **ListView**. These columns set up the test for entering first and last names from the ActiveData file into appropriate columns.

- Steps whose description starts with "Using Dialog" coordinate the test to the window and control used to enter data.

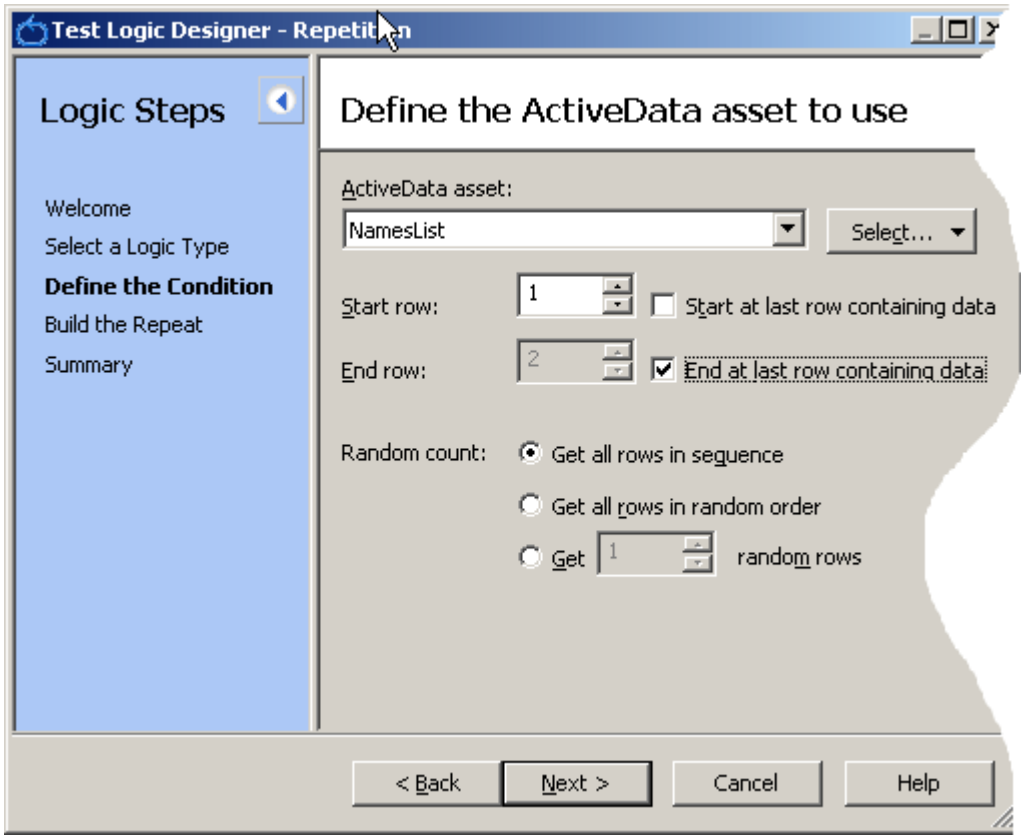
- Other steps in this sequence name the columns and add them to the **List view**.

Step 14: The `Repeat using activedata` step begins the repetition logic. The tester creates the step after recording the test. All steps between this step and the `End Repeat` step (step 23) repeat in the visual test for each name record in the ActiveData file. The repetition logic type used is **Repeating a sequence of steps using data from an ActiveData file**.

The **Start row**, **End row**, and **Random count** information entered while building the repetition logic determines the number of repetitions. Settings also determine the rows in the ActiveData file to use for ActiveData testing.

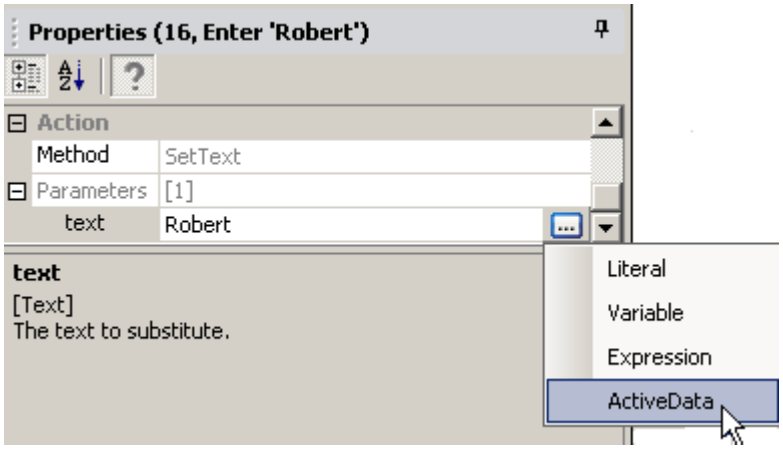
For this scenario, the tester uses all the rows in the data file in sequential order for ActiveData testing. Assuming the data file contains 10 rows of data and all are used, the number of repetitions for the repetition logic steps is 10.

The following shows the **Define the ActiveData asset to use** page of the **Test Logic Designer** with the settings used to create the repetition logic for this visual test.

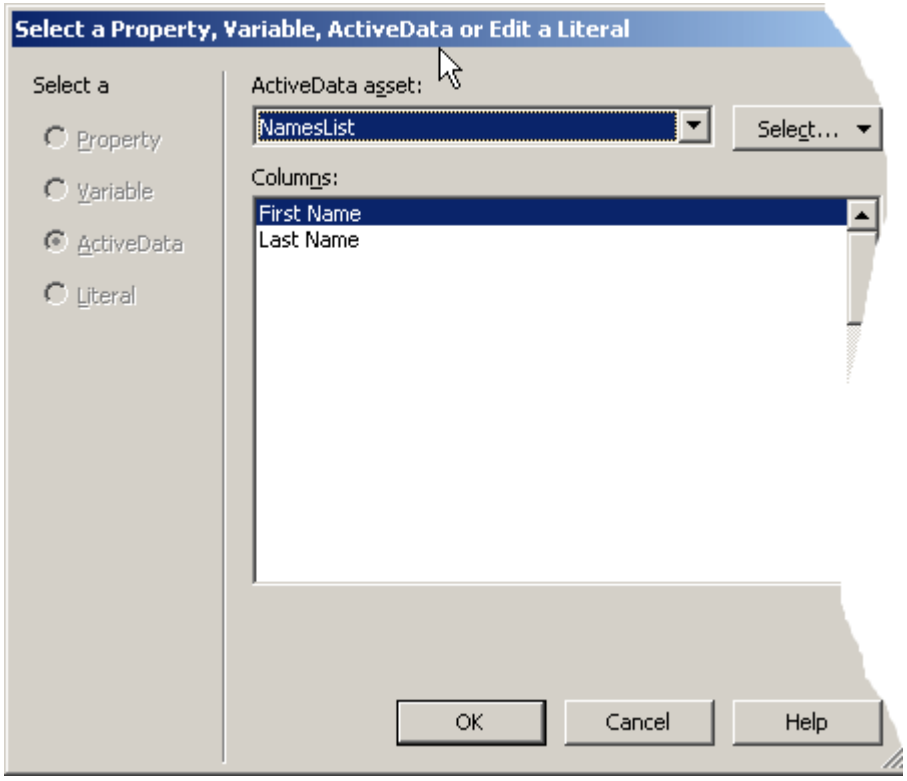


Steps 15-22: Recorded steps that have been modified after recording to use data from the ActiveData file. These steps enter data into the First Name and Last Name columns in the **ListView** control of the test application.

Modifying these steps to use data from the ActiveData file involves updating the input value to use an ActiveData value rather than a literal value. To do this, the tester clicks in the value area of the step's **text** property and selects **ActiveData** from the menu as shown in the following image.



This opens a dialog box where the tester selects the ActiveData asset containing the data file, and a column in the data file whose values are mapped to the represent the input value for the **text** property. In step 16 of this scenario, the tester replaces the literal text for the First Name in the Windows test application with data from the **First Name** column in the ActiveData file. The following graphic shows these settings in the dialog box.



This modifies the step text to show:

- The name of the ActiveData asset being used in the test.
- The type of data used (Text).
- The name of the column containing the data used in the step.

For more information about modifying automation test steps to use data from an ActiveData file, see *Mapping Data in an ActiveData File to Data in a Visual Test*.

Step 23: The `End Repeat` step created as part of the repetition logic for the visual test. This step closes the repetition logic loop.

Step 24: The <<End>> step. Concludes visual test playback.

Sample Visual Test: Using Decision Logic to Verify a Control Selection

This sample visual test verifies that a **ComboBox** control in a Windows sample test application has been selected. If the tested **ComboBox** had not had a selection since the time it was rendered, no text appears in the **ComboBox**. The tester uses decision logic to determine if the text of the **ComboBox** is null, and performs different actions based on the result.

Sample Visual Test: Using Decision Logic to Verify a Control Selection - Part 1 Scenario

Test Scenario

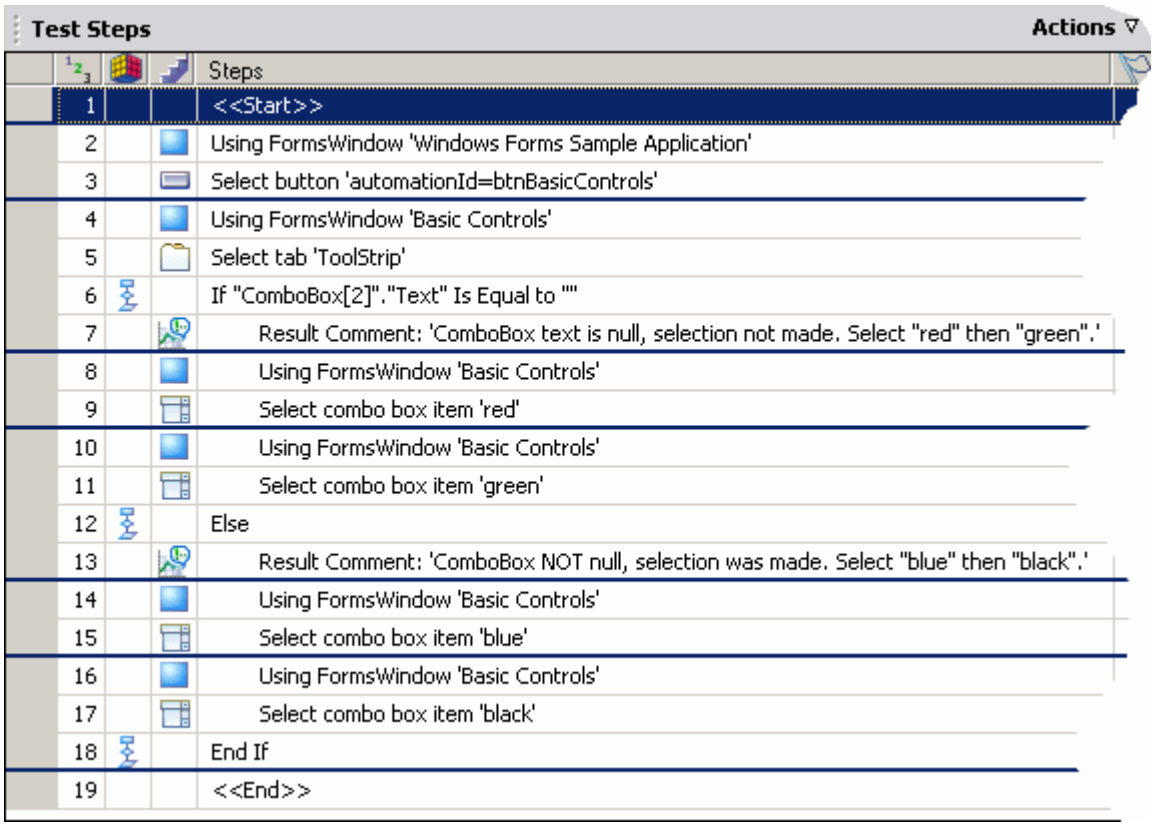
The tester wants to make a different set of selections against a **ComboBox** based on whether any value for the control has been previously selected. The tester wants one set of actions tested against the **ComboBox** if its existing text is empty (indicating it has not been selected), and a different action if not empty.

To do this, the tester sets up the visual test to evaluate whether the **Text** property of the control contains any text prior to performing actions against it. The decision logic uses the absence of text in the control's **Text** property to determine actions that will follow. The different set of actions based on the decision condition are part of the decision logic's If/Else construction. The tester also wants to report the outcome of the decision logic and the actions taken to the test results.

Sample Visual Test: Using Decision Logic to Verify a Control Selection - Part 2 Details

Sample Visual Test

The following shows the sample visual test steps in the **Test Steps** pane based on the test scenario and highlights the progression of steps in the test.



Step Details

Details of the visual test by steps is as follows:

Step 1: The `<<Start>>` step.

Step 2: A screen step that indicates the automation step that follows plays back using the specified recorded screen. When selecting the step in the **Test Steps** pane, the associated screen appears in the **Screen Preview**.

Step 3: A recorded step that clicks a **Button** control in the test application. Clicking the button displays a set of controls in various tabs. The **Toolstrip** tab contains the control being tested.

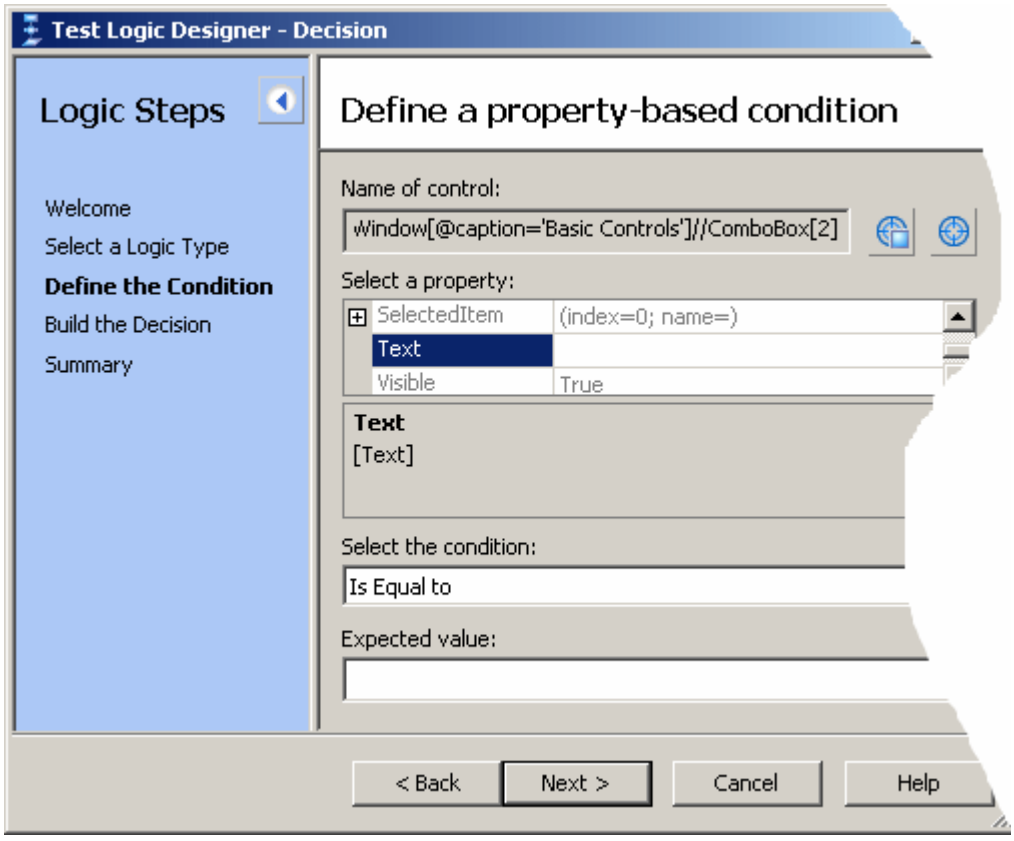
Steps 4-5: Recorded steps access and select the **Toolstrip** tab that contains the **ComboBox** control being tested.

Step 6: The `If` step that starts the decision logic, part of decision logic created by the tester after recording the test. The logic icon indicates that this step is a decision logic step.

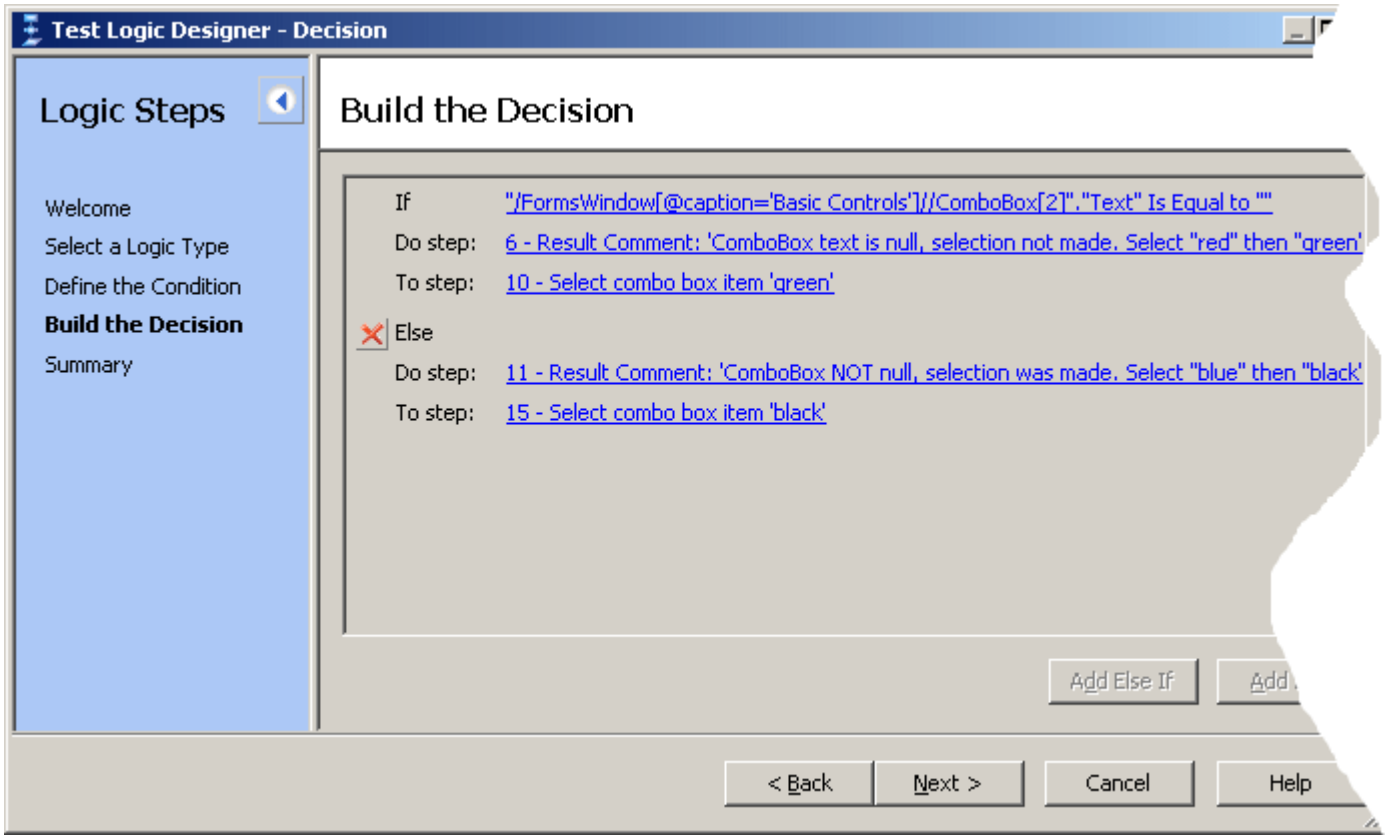
The decision logic evaluates the content of the **ComboBox** control's **Text** property. If no value has been set for the control, the **Text** property should be `Nothing` or blank. The visual test executes one set of steps if the value is `Nothing` or blank, and a different set of steps if the value is not `Nothing` or blank.

To set up the decision logic using the **Test Logic designer**, the user:

- Selects a Logic Type to **based on the property of a control**.
- Uses the define a property-based condition page of the **Test Logic designer** to do the following:
  - Identify the name of the control using the application under test.
  - Select **Text** in the **Select a property** List of the **Test Logic designer**.
  - Ensure that the condition evaluates to **is Equal to** and the **Expected Value** field is blank, as shown in the following graphic.



- Uses the **Build the Decision** page of the **Test Logic Designer** to specify the sequence of steps to run when depending on the conditional logic result. In this scenario, the user chooses to execute steps 6 through 10 of the result is true. The user then clicks **Add Else** to add the If step to the decision logic and chooses to execute steps 11 through 15 if the result is false, as shown in the following graphic.



Note: Step numbers subsequent to the `If` and `Else` steps change when these steps are inserted into the visual test. In this scenario, the following step numbers are based on their numbering after the decision logic has been added to the test, and not as shown in the previous graphic.

Step 7: A `Result Comment` step created by the tester. The step executes if the decision logic evaluates to true. If executed, the step sends a message to the test results indicating that the **ComboBox** contains no text and therefore has not been previously selected. It also indicates that available values in the control will be selected in subsequent steps.

Steps 8-11: These steps access and select different available **ComboBox** items if the decision logic evaluates to true. These steps first select `red` then `green` in the **ComboBox.**

Step 12: The `Else` step in the decision logic. The decision logic creates this step if **Add Else** is built into the logic. The `Else` step indicates that steps between it and the `End If` step in the decision logic are played back when the condition of the `If` step in the decision logic is not met.

Step 13: Another `Result Comment` step created by the tester. The step sends a message to the test results indicating that the **ComboBox** text is not `Nothing` and that available items will be selected.

Steps 14-17: These steps access and select different available **ComboBox** items if the decision logic evaluates to false. These steps first select `blue` then `black` in the **ComboBox**.

Step 18: An `End If` step created as part of the decision logic for the visual test. This step closes the `If...Else` logic.

Step 19: The <<End>> step. Concludes visual test playback.

Properties

This topic lists the properties for visual tests.

ActiveData Properties

The **ActiveData** property category includes the following properties:

Save Option Specifies whether to save updated ActiveData back to a specified ActiveData file immediately or not at all during playback.

Select one of the following values from the list:

- **Save data to ActiveData file now** – Any updated data from any specified ActiveData asset associated with the visual test is saved back to its respective ActiveData file when this step executes. ActiveData used in a visual test is typically updated through an expression. The asset containing the ActiveData file to which data will be saved is specified in the **ActiveData to Save** property.

  Note: Any ActiveData updated during the visual test playback is also saved to the respective ActiveData file when visual test playback completes, unless the visual test contains a step to cancel the save operation.
- **Do not save data to ActiveData file** – No updated ActiveData is save back to the ActiveData file for the ActiveData asset specified in the **ActiveData to Save** property.

Name Specifies an ActiveData asset containing the file whose updated data is either not saved at all during playback, or saved when this step executes, depending on the value selected for the **Save Option** property. For more information, see *Associating an ActiveData Assert With a Visual Test*.

Select the appropriate ActiveData asset from the list. The list contains ActiveData Assets that have been associated with the visual test.

Advanced Properties

The **Advanced** property category includes the following properties:

Buttons to display Determines the button or buttons to display in the message box.

Select the buttons to display from the list. The list includes the following:

- **OK** – Display the **OK** button in the message box.
- **OK, Cancel** – Display **OK** and **Cancel** buttons in the message box.
- **Abort, Retry, Ignore** – Display **Abort**, **Retry**, and **Ignore** buttons in the message box.
- **Yes, No, Cancel** – Display **Yes**, **No**, and **Cancel** buttons in the message box.
- **Yes, No** – Display **Yes** and **No** buttons in the message box.
- **Retry, Cancel** – Display **Retry** and **Cancel** buttons in the message box.

Default button Determines which button is automatically selected when the message box displays.

Select the **Default** button from the list. The list contents are based on the value selected for the **Buttons to display** property.

Icon to display Displays a selected icon in the message box. You can display an appropriate icon to indicates the message content. Possible values are:

- **(none)** – Do not display an icon in the message box. The is the default value.
- **Critical** – Display the **Critical** icon in the message box. Use this icon to alert the tester of a serious error or condition that requires attention.
- **Warning query** – Display the **Warning query** icon in the message box. Use this icon to alert the tester of a condition that requires a response.
- **Warning message** – Display the **Warning message** icon in the message box. Use this icon to alert the tester of an error or condition that does not require a response.

- **Information message** – Display the **Information message** icon in the message box. Use this icon when displaying information in a message box.

Assignment Properties

Use the **Assignment** properties to:

- Set an available property for a specified control to a variable so that the property value can be used more efficiently in the visual test. These properties appear for a `Put the [property] of the [control] to [variable]` step.
- Specify the local variable into which the global variable's value is stored in a `Get global` step.
- Define a global parameter in a `Set global` step.
- Specify eCATT script arguments to import from or export to an eCATT script.

Note: Properties that display in this category are based on the test step type. Some properties are not used for all step types that include this property category.

This category includes the following properties:

Global Variable Name

Specifies the name of a global variable to be created (in a `Set global` step) or retrieved (in a `Get global` step).

Type the desired name for the global variable. Valid characters for global variable names are uppercase and lowercase alphanumeric characters and the underscore ("ABC", "abc", "_"). Global variable names should be unique.

Tip: When retrieving a global variable that has been created in a `<<Start>>` step, its name must be known in advance. You can open the visual test where the global variable is set and get the name from the Global variable name property of the appropriate `Set global` step.

eCATT Argument Name

Specifies the name of an eCATT script argument to be imported from or exported to an eCATT script.

Type the name of the eCATT script argument whose value is to be imported or exported. The argument name must be defined in the argument container for the visual test using the **eCATT Argument Container** dialog box. A playback error occurs if the name entered does not exactly match the name defined in the argument container.

Property Name

Specifies the name of an available property for the control identified in the step.

The control identified for the step displays in the **Object Type** property of the **Properties** window. Select an available property from the list to set it's associated value to a variable.

Local Variable Name

Specifies the name of the local variable into which a specified value is stored.

Select a local variable from the list. Local variables must first be created in the visual test to display in the list.

For steps that set a property of a control to a local variable, the value of the selected property is stored in the specified local variable.

For steps that import eCATT script arguments, the value of the script argument is stored in the specified local variable.

Value

The value given to the global variable when initiated. Type in the value or right-click in the value area and assign a value.

Command Properties

This category includes the following properties:

Path Name Specifies the path and file name of the application or program to start.

Click **Select File** and use the **Select File** dialog box to navigate to and select the file that starts the desired program or application.

If the operation to perform is **Find** or **Explore**, only the a path name is required for this property.

Operation Specifies an action to perform when the application or program is started.

Select an action to perform from the list. The list includes the following:

- **Open** (default) – Opens the specified file using any named parameters. The file can be an application, a document file, or a folder.
- **Edit** – Starts an editor and opens the file for editing. If the file specified by the **Path Name** property is not a document file that can be opened in a document editor, the operation does not execute.
- **Explore** – Uses Windows Explorer to open the folder specified by the **Path Name** property.
- **Find** – Starts Windows Explorer in search mode starting from the specified directory. The directory can be specified using either the **Path Name** property or the **Directory** property. If using the**Path Name** property and including a file name, playing back the test does not execute the step.
- **Print** – Prints a document file specified by the **Path Name** property. If the file specified by the **Path Name** property is not a document file, playing back the test does not execute the step.

Parameters Specifies any parameters to be passed to the application for startup.

If the file specified by the **Path Name** property is an application file, use this property to specify a null-terminated string containing parameters passed to the application and used for startup. If the file specified by the **Path Name** property is a document or other type of file, leave the value blank.

Directory Specifies a default directory that can be used for the action specified by the **Operation** property.

Window Style Specifies how the application or program displays when it starts.

The value displays in the test step as an integer that represents the selected window style. Select the value from the list. Values include the following:

- **Hide** – Window is hidden when program or application starts and focus is passed to another window.
- **Maximize** – Maximizes the application window.
- **Minimize** – Minimizes the application window at startup and activates the next top-level window in the z-order.
- **Restore** – Starts and displays the application window. If the window is minimized or maximized, then it is restored to its original size and position. Use this value when restoring a minimized window.
- **Show** – Starts the application window and displays it in its current size and position.
- **Show default** – Sets the window display state based on the information provided by the program that starts the application.
- **Show maximized** – Starts the application and displays it in a maximized window.
- **Show minimized** – Starts the application and displays it in a minimized window.
- **Show minimized, do not activate** – Displays the application as a minimized window. The current active window remains active.
- **Show in current state, do not activate** – Displays the application in its current window state. The active window remains active.

- **Show in most recent position, do not activate** – Displays the application in its most recent window size and position. The active window remains active.
- **Show normal** – Starts the application and displays a window. If the window's most recent position minimized or maximized, the window is restored to its original size and position. Use this value when displaying the window for the first time.

Delay Properties

This category includes the following properties:

Delay type	Sets the delay interval for the step. Select **Seconds** or **Milliseconds** from the list.
Delay amount	Specifies the amount of time for the delay based on the delay type. Type a literal value or click **Select** to assign a variable, expression result, or ActiveData value.

Flag Settings Properties

The **Flag** settings property category includes the following properties:

Create flag on failure	Determines whether a flag is created on failure. To edit this property, set the value to either **True** or **False**. If set to **True**, a flag is created. If set to **False**, a flag is not created.
Flag description	Contains a description of the flag that is created on failure. The description appears in the results of the visual test and on the **Start Screen**. To edit this property, type the desired description for the flag.

General Properties

The **General** property category includes the following properties:

Title on message box	Determines the text that displays in the title bar of the message box. The default value is Silk Test. To change the default, type a value or click **Select** to assign a value as message box title.
Message text	Specifies the text content that displays in the message box. Virtually any information available to a visual test can be used in the message content. Type the message content directly or click **Select** to assign a value as message text. To combine literal text and other information such as data returned to a variable, click **Select** and select **Expression** from the menu. Use the **Expression Designer** to create the combined string. For example, to display descriptive text with a count of the number of items in a ComboBox, the expression might look similar to the following code: `"Message box text is: " + ToString (intCount)` Where `"Message box text is: "` is the descriptive text string + is the Operator used to combine strings `ToString (intCount)` is a variable that holds the number of items in the control `intCount` converted to a string using the `ToString ()` conversion function.

ActiveData	Displays the number of ActiveData assets whose data files are associated with the visual test. Associate active ActiveData assets with the visual test by clicking **Browse for ActiveData** in the value area to select assets using the **Browse for ActiveData** dialog box.
Item[n]	Displays the name of an ActiveData asset associated with the visual test.

Get Playback Setting Properties

The **Get Playback Setting** property category includes the following properties:

Playback setting	Controls an aspect of specific playback information help. For a `Put playback setting '[playback setting]' into '[local variable name]'` step, the value is a playback setting for which the current setting value is stored to a local variable. The value is retrieved from one of the following: • The current value for the playback setting as set in the playback options. • A defined run environment (when the **RunEnvironment** playback setting is set in a visual test). • A **Set playback setting** step in the visual test. Select the **Playback Setting** from the list.
Local variable name	Specifies the name of a local variable in which a value is stored. For a `Put playback setting '[playback setting]' into '[local variable name]'` step, the value of the selected playback setting is stored to the named local variable. Select a local variable from the list. Local variables must first be created in the visual test to display in the list.

Miscellaneous Properties

The **Miscellaneous** property category includes the following properties:

Description	Displays the description of the selected test step. To edit this property, type a description for the selected test step in the property value text box.
Last modified	Displays the date and time of when the test step was last modified. This property is read-only.

Object Properties

The **Object** property category includes the following properties:

Class Name	Identifies the class name for the selected object.
Locator	Identifies the object to test. You can identify the object that you want to test using the **Screen Preview** pane or the actual application. Or, you can select a literal value, expression, ActiveData, or a variable to use as the locator.
Property	Returns a list of properties that can be retrieved for the specified object and assigned to a variable. Select a property from the list. Or, you can select a literal value, expression, ActiveData, or a variable to use as the property.
Local variable name	Specifies the name of the local variable into which the specified value is stored.

On Error Go To Properties

The **On error go to** property category includes the following properties:

Action to take Sets the action to take when an error occurs during playback of a visual test. Select one of the following options from the list:

- **Show default playback error dialog** – Displays the **Playback Error** dialog box when a playback error occurs, which is used to determine a playback action to take when an error occurs. Use the **Playback Error** dialog box to enter debug mode to diagnose errors in a visual test.

  Displaying the **Playback Error** dialog box is the default behavior when playback encounters an error.
- **End playback** – Ends visual test playback when an error occurs. No other steps in the visual test play back. When playback ends, the **Playback Complete** dialog box appears, which determines the action to take when playback completes.
- **End test** – Plays back the visual test specified in the **Asset** property, then ends at the step where the error occurred. When playback ends, the **Playback Complete** dialog box appears, which determines the action to take when playback completes.
- **Retry step** – Plays back the visual test specified in the **Asset** property, then attempts to execute the step that caused the error. If no visual test is specified in the **Asset** property, then playback simply attempts to re-execute the step that caused the error. You can specify the number of replay attempts in the **Number of retries** property. The default number is 5. If the step execution is not successful when the specified number of retries is reached, a playback error is generated.

  Note: Silk Test Workbench does not generate a playback error while attempting to re-execute a step. If you set the **Number of retries** property to 0, the step is retried forever. This could cause Silk Test Workbench to appear to not be responding during a playback while continually retrying a failed step.

  The **Retry step** action is useful for projects that have visual tests that reliably handle errors or fix potential errors to anticipated test issues. For example, if a visual test accesses a page in a browser application that may not reliably load, a 'fail to attach' error may occur. The **Retry step** action can direct the visual test to another visual test that reliably loads the page before attempting to interact with it, thereby resolving the playback error.
- **Resume next** – Plays back the visual test specified in the **Asset** property, then attempts to execute the step immediately after the step that caused the error. The step that caused the error is not re-executed.
- **Go to** – Plays back the visual test specified in the **Asset** property, then navigates to the predefined Label step specified in the **Label** property and continues playback. Using **Go to** allows that visual test to skip steps that could potentially contain failures similar to the one that occurred during playback, and navigate to another section of the test.

  Note: If a specified label step used for error handling is placed before the step where the error occurs, the error continues to occur because the affected step continues to be played back in a loop.

Asset Specifies the visual test to run when an error occurs during playback of a visual test. To select a different visual test, click **Browse for Visual tests**, and then select a visual test.

Tip: Once you have specified the asset and created the step, another action can be selected in the **Action to take** property to change the behavior of the step. For example, if the **Action to take** property is set to **End test** and an asset selected for the **Asset** property, the step text is as follows:

```
On error playback '[visual test]' and then 'End test'
```

The **Action to take** property value can now be changed to change the behavior of the step. Changing that to value to **Retry step** changes the step as follows:

```
On error playback '[visual test]' and then 'Retry step'
```

Label Contains a list of available label steps the visual test can navigate to if an error occurs. Select a label step name from the list.

This property only appears when **Action to take** property is set to **Go to**.

Playback Settings for a Visual Test

You can use the following playback settings to configure the playback behavior of a visual test:

Application Ready Timeout Specifies the number of milliseconds to wait for a newly launched application to become ready. If the application is not ready within the specified timeout, Silk Test Workbench raises an exception.

Buttons to close windows Specifies the buttons used to close windows with the `CloseSynchron` method.

Buttons to confirm dialogs Specifies the buttons used to close confirmation dialog boxes that appear when closing windows with the `CloseSynchron` method.

Close unresponsive applications Specifies whether unresponsive applications are closed. An application is unresponsive if communication between the Agent and the application fails, e.g. times out.

Close window timeout Specifies the number of milliseconds to wait before the next close strategy is tried. The Agent executes four close attempts before failing, so the total time before a close fails is four times the value you specify.

Control capture Enables Silk Test Workbench to capture controls and their properties during playback. If the control capture setting in the Record Options is set to No, this option is overridden and controls are captured during playback.

Note: If the **Screen capture** setting is set to **None** for recording, you must set **Control capture** to **Yes** and the playback **Screen capture** setting to a value other than **None** to capture controls for playback.

Ensure object is active Ensures that the target object is active.

Highlight object during playback Specifies whether the current object is highlighted during playback.

Keyboard event delay Sets the delay in milliseconds between playback of keyboard strokes.

Be aware that the optimal number you select can vary, depending on the application that you are testing. For example, if you are testing a Web application, a setting of 1 millisecond radically slows down the browser. However, setting this to 0 (zero) may cause basic application testing to fail.

Keys to close dialogs Specifies the keystroke sequence to close dialog boxes that open after trying to close a window with the `CloseSynchron` method. Examples include: <ESC>, <Alt+F4>.

Menu items to close windows Specifies the menu items used to close windows with the `CloseSynchron` method. Examples include: "File/Exit*", "File/Quit*".

Mouse event delay Specifies the delay in milliseconds used before each mouse event.

Object enabled timeout

Specifies the time in milliseconds to wait for an object to be enabled during playback. As soon as the object is enabled, Silk Test Workbench can interact with it. For example, if the object is a button, Silk Test Workbench can click it when it is enabled.

Object resolve timeout

Specifies the time in milliseconds to wait for an object to be resolved during playback. As soon as the object is resolved, Silk Test Workbench can recognize it.

Playback mode

Defines how controls are replayed. Use low level to replay each control using the mouse and keyboard. Use high level to use the API to replay each control. All controls have a default playback mode assigned. When the default replay mode is selected, each control uses its default playback mode. The default mode delivers the most reliable results. Selecting low or high level playback overrides the playback mode of all controls with the playback mode selected.

Result pass criteria (percentage)

Sets the overall criteria to define the success of future runs. For example, a result pass criteria of 90% means that at least 9 out of 10 verifications must pass for the result of the playback to pass.

Save all information

Specifies whether all information for a visual test is saved after playback finishes.

Screen capture

Determines how Silk Test Workbench captures test application screens during playback.

Select from the following options:

- **Same as Recorded** – Use the same setting as the **Screen capture** setting that existed in the **Record** options when the visual test was recorded.
- **Active Window** – Capture only the test application's active window during recording.
- **Application** – Capture the test application and any windows within the test application.
- **Desktop** – Capture screens of all applications, including the test application, that are visible in the desktop during recording.
- **None** – Captures no screens during recording.

Note: When this setting is set to **None**, no controls or screens are captured in the **Results** window, regardless of the values specified in the **Control capture** and **Screen capture (test steps)** settings.

Screen capture (test steps)

Determines whether Silk Test Workbench captures individual screens for the test application during recording. This enables you to view the actions that occur on individual screens in **Screen Preview** in conjunction with individual test steps in the **Test Steps** pane. Additionally, you can compare the screens that are recorded during playback against the screens captured when the visual test was first recorded if test step screens were captured during recording also.

Select from the following options:

- **Same as Recorded** – Use the same setting as the **Screen capture (test steps)** setting that existed in the **Record** options when the visual test was recorded.
- **Yes** – Capture screens during recording. Use the **Screen capture** setting to determine how the screens are captured.
- **No** – Captures no individual test steps during playback. If the **Screen capture** setting is set to anything other than **None**, the group screen associated with the individual test step is displayed in **Screen Preview**.

Screenshot delay	Specifies the number of milliseconds to wait once focus has been set to a new window before taking a screen shot of the window.
Synchronization exclude list	Specifies the URL for the service or Web page that you want to exclude during page synchronization. Some AJAX frameworks or browser applications use special HTTP requests, which are permanently open in order to retrieve asynchronous data from the server. These requests may let the synchronization hang until the specified synchronization timeout expires. To prevent this situation, either use the HTML synchronization mode or specify the URL of the problematic request in the **Synchronization exclude list** setting. Separate multiple URLs with a comma. Type the entire URL or a fragment of the URL, such as *http://test.com/timeService* or *timeService*.
Synchronization mode	Configures the supported synchronization mode for HTML or AJAX. Using the HTML mode ensures that all HTML documents are in an interactive state. With this mode, you can test simple Web pages. If more complex scenarios with Java script are used, it might be necessary to manually script synchronization functions, such as `WaitForObject`, `WaitForProperty`, `WaitForDisappearance`, or `WaitForChildDisappearance`. Using the AJAX mode eliminates the need to manually script synchronization functions.
Synchronization timeout	Specifies the maximum time in milliseconds for an object to be ready.
Unresponsive application timeout	Specifies the timeout in milliseconds for canceling pending playback actions.

Playback Settings for an Inserted Script or Visual Test

Playback settings determine the playback settings to be used for the visual test or script being executed within the visual test. Select one of the following values from the list:

Use system defaults	Use the defaults for each playback option as it appears in the **Options** dialog box.
Inherit settings from parent	Use the playback settings of the parent visual test for the current playback run.
Copy settings from parent	Use the playback settings of the parent visual test for the current playback run and for all future playback runs.

Result Properties

The **Result** properties category includes the following properties:

Steps	Displays the name of the selected test step in the **Test Steps** pane.
Result	Displays the result of the selected step in the **Test Steps** pane.
Result detail	Displays the result details of the selected step in the **Test Steps** pane.
Line number	Displays the line number of the selected step in the **Test Steps** pane.
Command details	Displays the log details of the selected step in the **Test Steps** pane.
Date/Time	Displays the time and date of the selected step in the **Test Steps** pane.
Milliseconds	Displays the milliseconds of the selected step in the **Test Steps** pane.
Name	Displays the asset name of the selected step in the **Test Steps** pane.

User name	Displays the name of the user who played back the visual test.
Machine name	Displays the name of the computer on which the visual test was played back.

Set Playback Setting Properties

The **Set Playback Setting** property category includes the following properties:

Playback Setting	Select the **Playback** setting from the list.
Value	Assigns a value to the selected playback setting. The assigned value is used only for the visual test containing the `Set playback setting` step. Depending on the selected playback setting, select the value from a list, or click **Select** to assign a value as the playback setting value.

Timer Properties

The **Timer** properties category includes the following properties:

Timer Action	Sets the action of the timer test step. • **Start** – Starts the timer. When the timer stops, the timer is reset to 0. If you try to start a timer that is already running, an error occurs. • **Stop** – Stops the timer. The timer retains the amount of time and you can restart it using the **Resume** action if necessary. • **Resume** – Restarts a timer that was stopped using the **Stop** action. The timer resumes counting from the amount of time that was recorded previously. • **Reset** – Resets the time to 0.
Timer Number	Sets the timer number. When timing multiple conditions, you can differentiate the timers by assigning a unique number for each timer. Select a timer number from the list. You can include up to 10 timers in a visual test. This property is only available when verifying the time taken to run a sequence of steps.

Verification Result Text Properties

The **Verification result text** property category includes the following properties:

Verification passed text	Contains the message to display in the result description of a visual test when the specified verification logic condition passes. To edit this property, select the property and type the desired description.
Verification failed text	Creates the message to display in the result description of a visual test when the specified verification logic condition fails. To edit this property, select the property and type the desired description.

Wait Properties

The **Wait** properties category includes the following properties:

Locator	Specifies which object the step waits to exist or disappear. Identify the control using any of the following methods: • **Application Under Test** – Click this button to identify a visible control directly from the application under test.

- **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
- **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.
- **Select** – Click this button to assign a literal, variable, expression result, or ActiveData value.

Wait type Specifies whether to wait for the object to exist or disappear.

Timeout Sets the maximum number of milliseconds to wait for the object to exist or disappear. To edit this property, select the property and type a number. By default, this value is set to the **Default wait timeout (milliseconds)** option value. If no object matches within the timeout period, a playback error occurs.

Creating Scripts

Silk Test Workbench uses scripts in the same manner as visual tests to mimic the actions that are performed while testing an application. It controls an application or Web page in the same way that a user would by using keystrokes and mouse actions to select menus, list items, and buttons. During recording, Silk Test Workbench generates all of the keystrokes and mouse clicks that test the application.

Silk Test Workbench's scripting language is Microsoft's Visual Basic, a robust programming language that gives you total control over any application running in the Microsoft .NET framework. .NET scripts contain the functionality of a high-level programming language as well as features designed specifically for software control and testing. Using the language, you can develop scripts that:

- Run automatically at a specified time of day.
- Run entirely unattended.
- Interact with users to receive confidential information such as IDs and passwords.

The actions taken to test an application, making menu selections, typing data, checking the way it is processed, and so on, are represented in scripts as VB.NET commands. These commands are inserted into a script and can be modified and played back.

Silk Test Workbench lets you quickly record and playback scripts. Modify scripts to include “hand-coded” statements that cannot be recorded, to make amendments to reflect changes in the test application, or to create new scripts by cutting and pasting code from existing scripts. You can also use the **Identify Object** dialog box to record locators or object map items for individual objects and then use the locator or object map item in a script.

Recording is the best way to quickly create scripts. Recording enables you to become familiar with how Silk Test Workbench generates code for scripts. This provides familiarity with the basics of script creation, which then allows better understanding of creating efficient test solutions, modularizing scripts, and creating independent code modules.

Benefits of Using Scripts

Scripts record the actions at an object level that you use to test an application. These actions include making menu selections, typing data, clicking icons, and verifying results. Maintaining test procedures in scripts has several advantages:

- Scripts enable power and flexibility because they are highly extensible. Users can develop highly complex solutions for complex test cases with the power of .NET scripting.
- With scripts, users can leverage the functionality provided by external assemblies to enhance the functionality available within scripts.

- Developers and testers share a common scripting language, VB.NET.
- If the test application changes, scripts do not have to be rewritten from scratch.
- New test procedures can easily be built by copying and modifying existing scripts.
- Scripts facilitate a modular approach to testing and are highly re-useable. Scripts can call other scripts or functions.
- Scripts can "loop" to test a process over and over.
- Scripts provide quick building of intelligent tests that verify expected results and handle unexpected situations.
- Scripts can document test processes.
- Scripts can be built before the test application is complete.

Best Practices for Creating Test Scripts

The way in which you write your test cases might have a great impact on the performance and stability of your test set. During recording, Silk Test Workbench creates scripts that are as fast and stable as possible. However, there might be circumstances that require you to manually create or edit test scripts. This topic provides some general guidelines that might help you create test scripts that are maintainable, reusable, and lead to stable tests.

- Name your tests consistently and ensure that test names are self-explaining. Try to make the names correspond with the application under test and the tested functionality. For example, the test names *MyApp_SuccessfulLogin* and *MyApp_FailingLogin* are far easier to understand for other users than *Untitled_42* and *Untitled_43*.
- Describe your test cases as thoroughly as possible in a comment. Without a good description of the test case in natural language, someone who needs to change the implementing code might not be able to comprehend what exactly the test is doing.
- Ensure that your application under test is at the proper state when the test case starts. Return the application under test to the correct state before executing the actions in a test case.
- Ensure that your application under test is at a proper state when the test case finishes. If additional tests depend on the outcome of the test, ensure that they can start. Return the application under test to the correct state when the actions in a test case are executed.
- Whenever possible, ensure that your test cases are not depending on the results of other test cases. When this is not possible, ensure that the test cases are executed in the right order.
- Add verifications to your tests, to test the correctness of your application under test as well as the functional flow.
- Use keyword-driven testing to create highly reusable action sets. Bundle commonly used actions into keywords, combine keywords that are often executed sequentially into keyword sequences, and execute combinations of keywords and keyword sequences as keyword driven-tests.
- To keep your tests maintainable and reusable, prefer writing multiple simple test cases that are combinable to writing complex test cases.
- To avoid redundancies in your test set, prefer updating existing test cases to adding new test cases.

Script Syntax

Understanding the syntax of recorded script lines makes scripts easier to read. Commands that test objects are composed of identical syntax elements. The syntax elements include a `With` statement for the application windows, such as the main application window or dialog boxes, followed by a locator or object map item that identifies the class, attribute, and action.

Using Object Maps

An object map is a test asset that contains items that associate a logical name (an alias) with a control or a window, rather than the control or window's locator.

By default, Silk Test Workbench includes object map items in the script context when you record a script.

The following example shows a typical recorded action in a script that tests a Web application.

```
With _desktop.BrowserApplication("webBrowser")
  With .BrowserWindow("browserWindow")
    .DomListBox("quickLinkJumpMenu").Select("Auto Quote")
  End With
End With
```

The `With_ desktop.<application>` portion identifies the main application window.

The `DomListBox` portion in the previous example identifies the class to use.

The `("quickLinkJumpMenu")` portion identifies the attribute for the object. In this case, the attribute identifies the list box link menu. While the `Select()` portion identifies the action or command to perform against the object.

After the initial `With` command is defined, you do not need to repeat it in additional calls for that window. For each additional window that you test, you must specify a `With` statement. For example, the following code shows how multiple windows are called within the same script:

```
With _desktop.Window("untitledNotepad")
  .MenuItem("aboutNotepad").Select()
    With .Dialog("aboutNotepadDialog")
      .PushButton("ok").Select()
    End With
End With
```

Object map items are enclosed in parentheses and quotation marks ("") and replace the need to use locator captions.

Using Locators

Within Silk Test Workbench, literal references to identified objects are referred to as *locators*. By default, Silk Test Workbench includes object map items in the script context when you record a script. If you turn off object maps, locators are included in the script context instead of object map items when you record a script.

The following example shows a typical recorded action in a script that tests a Web application.

```
With _desktop.BrowserWindow("/BrowserApplication[1]//BrowserWindow")
  .DomLink("@textContents='Court: Gender pay lawsuit can go to
trial'").Select()
End With
```

The `With_ desktop.<application>` portion identifies the main application window.

The `DomLink` portion in the first example and the `MenuItem` portion in the second example identifies the class to use.

The `("@textContents='Court: Gender pay lawsuit can go to trial'")` portion identifies the attribute for the object. In this case, the attribute identifies the text content. While the `Select()` portion identifies the action or command to perform against the object.

After the initial `With` command is defined, you do not need to repeat it in additional locator strings for that window. For each additional window that you test, you must specify a `With` statement. For example, the following code shows how multiple windows are called within the same script:

```
With _desktop.Window("@caption='Untitled - Notepad'")
  .MenuItem("@caption='About Notepad'").Select()
    With .Dialog("@caption='About Notepad'")
      .PushButton("@caption='OK'").Select()
    End With
End With
```

The caption attribute identifies the objects to test. The caption identifies the main application window `("@caption='Untitled - Notepad'")` followed by the menu item `("@caption='About`

`Notepad'")`, the subsequent dialog box `("@caption='About Notepad'")`, and the button `("@caption='OK'")` to click.

Driver Scripts

A *driver script* triggers and monitors other scripts to test a test application. A driver script typically performs no testing of its own, although it may use conditional and other control logic to manage optional execution of other tests within the driver.

A complete test suite contains many individual tests, executed at different points within the test application. Each script in a test suite can be invoked by a driver script. Driver scripts are built in the same way as scripts, but the driver script does not actually test the test application. It merely “drives” the test application to the proper point to begin a test and then calls one or more scripts to begin testing at that test site. When the test finishes, the driver script regains control and proceeds to the next test site. You can create a driver script in the same way you manually create individual scripts.

Driver scripts facilitate a modular approach to application testing. Instead of creating one script that performs a complete test, you can create reusable scripts to cover individual parts of a whole test flow, then use a driver script to call each part. If a part of the overall test changes, only the script requiring the change needs to be updated. No change to the driver script is needed.

It is important to note that well-designed scripts return the test application to the original test site before completion. This ensures that:

- Driver scripts can always pick up from where they left off.
- You can add or remove tests at a test site without modifying the driver path.

Creating a Driver Script

Driver scripts facilitate a modular approach to application testing. Instead of creating one script that performs a complete test, you can create reusable scripts to cover individual parts of a whole test flow, then use a driver script to call each part.

1. Record a new script.
   For example, this script might start the test application and open a dialog box or window. If you are testing a Web application, the driver script might start the browser and open a specific Web page.
2. Stop recording when you reach the point at which you want to playback another script.
3. In the **Code** window, move the cursor to a new line and type:
   ```
   Workbench.RunScript ("script")
   ```
   where *`script`* is the name of the script.

   A well-designed script returns the test application to the original test site before completion to ensure that driver scripts can always pick up from where they left off.

   Scripts stored in Oracle databases are case sensitive. Any included script from an Oracle database must match the name in the database exactly or Silk Test Workbench fails to load the script.
4. If you want to play back scripts at additional test sites in the test application, continue to record steps to drive the test application to the appropriate state. Stop recording and insert another `RunScript` command.
5. When you have inserted all the tests that you want to include, record steps to close the test application or to return the test application to a state appropriate for other tests or test suites you may want to run.

Manual Scripting

While recording provides a quick and effective way to create scripts, there are often tweaks and enhancements needed to make scripts reusable, more reliable, and more effective. You can manually add code to any script to accomplish this. Since Silk Test Workbench uses VB.NET as its scripting language, some knowledge of VB.NET is needed. Silk Test Workbench extends the VB.NET scripting language to

provide robust and powerful scripting capabilities. See the *Silk Test Workbench Language Reference* for information on classes, methods and properties and how to use each in scripts.

Silk Test Workbench also provides other features and leverages VB.NET scripting features to make manual scripting easier. These features include:

- **Identify Objects** dialog box – The **Identify Objects** dialog box enables you to view how locators are recorded and which properties are available for those objects.
- Auto-completion and Syntactical Assistance – These facilitate manual coding. Using auto-completion and syntactical assistance, you can automatically access, display, and automatically enter VB.NET language elements, including keywords, enumerations, Boolean values, code syntax, available methods, and available VB.NET properties for any object.

Note: When you are manually creating a new test script, you have to add statements to your script to import the extensions for the classes that you use. For example, for Win32 methods, you have to insert the following line to the start of your script:

```
Imports SilkTest.Ntf.Win32
```

When you record a click on an accessible control, the import statements are automatically inserted into the script. Common Silk Test classes, for example `BaseGuiTestObject`, do not need to be imported because they are always available.

Recording a Script

Use Silk Test Workbench to record the actions performed when testing applications. When an action is performed, the action, and the responses of the application worked with, are recorded as commands in a script.

During recording, Silk Test Workbench records all interactions with the test application (except interaction with Silk Test Workbench itself) until recording is stopped. After you have finished recording, you can modify the script you have generated to add and remove steps.

1. Choose **File** > **New**. The **New Asset** dialog box opens.
2. Select **.NET Script** from the asset types list, and then type a name for the script in the **Asset name** text box.

   Note: The script is created in the Common project by default, but you can create it in another project by selecting it from the **Project** list.

   When using Silk Test Workbench with an Oracle database, all saved script names are case sensitive. Which means that a script named *SCRIPT1* can coexist in the same Oracle database with a script named *Script1* and not be considered a duplicate. However, when using Silk Test Workbench with either SQL Server or Access, saved scripts are not case sensitive.
3. *Optional*: Click **Options** if you want to change Silk Test Workbench record options.
4. Check the **Begin Recording** check box to start recording immediately.
5. Click **OK** to save the script as an asset and begin recording.

   If you click **OK** without checking the **Begin Recording** check box, Silk Test Workbench saves the script as an asset and displays the script template.

   The **Select Application** dialog box opens.
6. If you have not set an application configuration for the current project, select the tab that corresponds to the type of application that you are testing:
   - If you are testing a standard application that does not run in a browser, select the **Windows** tab.
   - If you are testing a web application or a mobile web application, select the **Web** tab.
   - If you are testing a native mobile application, select the **Mobile** tab.
7. To test a standard application, if you have not set an application configuration for the current project, select the application from the list.

8. To test a web application or a mobile web application, select one of the installed browsers or mobile browsers from the list.
   a) Specify the web page to open in the **Enter URL to navigate** text box. If an instance of the selected browser is already running, you can click **Use URL from running browser** to record against the URL currently displayed in the running browser instance.
   b) *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
   For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.
   c) *Optional:* Select an **Orientation** for the browser window.
   d) *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.
9. To test a native mobile application (app):
   a) Select the mobile device, on which you want to test the app, from the list.
   b) Click **Browse** to select the app file or enter the full path to the app file into the **Mobile app file** text field.

      Silk Test Workbench supports HTTP and UNC formats for the path.

      Silk Test Workbench installs the app on the mobile device or emulator.
10. Click **OK**. Silk Test Workbench minimizes and the application and the **Recording** window opens.
11. Record the actions that you want to test.

    During recording the Silk Test Workbench icon on the task bar flashes. For information about the actions available during recording, see *Actions Available During Recording*.
12. If you are recording against a web application on one of the supported browsers, and you want to interact with a control that you cannot access directly in the UI, for example because the control is hidden by another control, click **Toggle Hierarchy View** in the **Recording** window and select the control from the control hierarchy tree.

    This functionality is available if you are testing against one of the following browsers:

    - Microsoft Edge.
    - Apple Safari.
    - Mozilla Firefox 41 or later.
    - Google Chrome 50 or later.
    - A mobile browser.
13. *Optional:* Add verification logic to the test.

    - If you are testing a standard or web application, click `Ctrl+Alt` to add verification logic to the test. Silk Test Workbench temporarily suspends recording and displays the **Test Logic Designer** wizard.

      Follow the wizard through the process and click **Finish** to close the wizard and continue recording.
    - If you are testing a web application on Microsoft Edge or a mobile application, click on the object that you want to verify and click **Add Verification** in the **Choose Action** dialog box.

    For additional information about adding a verification, see *Adding a Verification to a Script while Recording*.
14. Stop recording by pressing `Alt+F10`, clicking **Stop** in the **Recording** window, or clicking the Silk Test Workbench taskbar icon. The **Recording Complete** dialog box opens. If the **Do not show this message again** check box is checked in the **Recording Complete** dialog box, this dialog box does not appear after recording is stopped. In this case, the script displays.
15. Perform one of the following steps:

    - Click **Playback** to close the **Recording Complete** dialog box and save the script.

      The **Playback** dialog box opens, which allows you to specify result information and playback the script.

- Click **Go to .NET Script** to open the script in the **Code** window.
- Click **Save** to save the script and exit the **Recording Complete** dialog box.

  The script appears in the **Code** window.

When the recorded test displays in the **Code** window, you can play it back at any time.

Recording a Script that Tests Multiple Test Applications

You can record a script that tests multiple test applications. For example, if you are testing an application that modifies a database and you use a database viewer tool to verify the database contents, you must add an additional application configuration for the database viewer tool.

1. Record a script or create one manually for the primary application that you want to test.
2. In the **Properties** pane, right-click and choose **Add Application Configuration**. The **Select Application** dialog box opens.
3. If you have not set an application configuration for the current project, select the tab that corresponds to the type of application that you are testing:
   - If you are testing a standard application that does not run in a browser, select the **Windows** tab.
   - If you are testing a web application or a mobile web application, select the **Web** tab.
   - If you are testing a native mobile application, select the **Mobile** tab.
4. To test a standard application, if you have not set an application configuration for the current project, select the application from the list.
5. To test a web application or a mobile web application, select one of the installed browsers or mobile browsers from the list.
   a) Specify the web page to open in the **Enter URL to navigate** text box. If an instance of the selected browser is already running, you can click **Use URL from running browser** to record against the URL currently displayed in the running browser instance.
   b) *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
   For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.
   c) *Optional:* Select an **Orientation** for the browser window.
   d) *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.
6. To test a native mobile application (app):
   a) Select the mobile device, on which you want to test the app, from the list.
   b) Click **Browse** to select the app file or enter the full path to the app file into the **Mobile app file** text field.

      Silk Test Workbench supports HTTP and UNC formats for the path.

      Silk Test Workbench installs the app on the mobile device or emulator.
7. Click **OK**.
8. Record additional steps for the script using the new application configuration.

Configuring a Script to Launch an Application that Uses the Java Network Launching Protocol (JNLP)

Applications that start using the Java Network Launching Protocol (JNLP) require additional configuration in Silk Test Workbench. Because these applications are started from the Web, you must manually configure the application configuration to start the actual application as well as the application that

launches the "Web Start". Otherwise, the visual test will fail on playback unless the application is already running.

1. If the test fails, because Silk Test Workbench cannot start the application, edit the application configuration.
2. Open the script in which you want to include the code to start the application.
3. In the **Properties** pane, navigate to the **Application Configurations** category.
4. Double-click on the application configuration that you want to modify. The **Edit Application Configuration** dialog box appears.
5. Edit the base state to ensure that the Web Start launches during playback.
   a) Click **Edit**.
   b) In the **Executable Pattern** text box, type the absolute path for the `javaws.exe`.
      For example, you might type:
      ```
      %ProgramFiles%\Java\jre6\bin\javaws.exe
      ```
   c) In the **Command Line Pattern** text box, type the command line pattern that includes the URL to the Web Start.
      ```
      "<url-to-jnlp-file>"
      ```
      For example, for the SwingSet3 application, type:
      ```
      "http://download.java.net/javadesktop/swingset3/SwingSet3.jnlp"
      ```
   d) Click **OK**.
6. Click **OK**. The test uses the base state to start the web-start application and the application configuration executable pattern to attach to javaw.exe to execute the test.

When you run the test, a warning states that the application configuration EXE file does not match the base state EXE file. You can disregard the message because the test executes as expected.

Configuring a Script to Launch in a New Browser Window

At times you may want to launch a separate browser window for a script rather than using the existing browser window. For instance, when you execute a Silk Test Workbench script from Silk Central Test Manager, the script uses the existing Silk Central window. To return to Silk Central, the user must click **Back** in the browser window. To avoid this behavior you can configure your script to launch a separate browser window for tests that you will execute from Silk Central.

1. Open the script in which you want to include the code to start the application.
2. If your script uses a base state, perform the following steps:
   a) In the **Properties** pane, navigate to the **Application Configurations** category.
   b) Double-click on the application configuration that you want to modify. The **Edit Application Configuration** dialog box opens.
   c) Uncheck the **Execute base state** check box.
3. Type `Imports System.Diagnostics` at the beginning of the script.
4. To define the application as a new process, type:
   ```
   Dim applicationProcess As New Process()
   ```
   where *applicationProcess* is the name of the application that you want to start.
   For example, to start Internet Explorer, type:
   ```
   Dim ieProcess As New Process()
   ```
5. Define the full path to the application by typing the following code:
   ```
   Dim psi As New ProcessStartInfo()
   psi.FileName = "applicationexecutable.exe"
   ```

where *applicationexecutable.exe* is the name of the executable file that launches your test application. For example to start Internet Explorer, type:

```
Dim psi As New ProcessStartInfo()
psi.FileName = "C:\Program Files\Internet Explorer\iexplore.exe"
```

Note: If the executable for the application that you want to test is in the same directory as Silk Test Workbench, you do not need to specify the full path.

6. To open the Web application that you are testing, set the `ProcessStartInfo.Arguments` property to the URL.
   For example, to start Internet Explorer and test the sample Web application, add the following line after the `Filename` line:

   ```
   psi.Arguments = "http://demo.borland.com/InsuranceWebExtJS/index.jsf"
   ```

7. Specify the code to start the process.
   For example, to start `ieProcess`, type:

   ```
   ieProcess.StartInfo = psi
   ieProcess.Start()
   ```

8. Save the script.

Identifying a Control in the Test Application (.NET Script)

When you record a .NET script, Silk Test Workbench automatically identifies controls. Identify a control name in the application that you are testing to change an existing control or to manually insert an action.

1. Click **Identify Object** in the toolbar. The **Identify Object** dialog box opens.
2. Click **Start Identify**.
3. Move the cursor to the application that you are testing. Controls appear highlighted as the cursor passes over them in the application. The related locator string or object map item shows in the **Selected Locator** text box.
4. Click the control that you want to use as it is highlighted.
5. *Optional:* Click **Show Locator Details** to display the locator tree and any related attributes in the **Locator Attribute** table.
6. *Optional:* You can replace a recorded locator attribute with another locator attribute from the **Locator Attribute** table.

   For example, your recorded locator might look like the following:

   ```
   /BrowserApplication//BrowserWindow//input[@id='loginButton']
   ```

   If you have a `textContents Login` listed in the **Locator Attribute** table, you can manually change the locator to the following:

   ```
   /BrowserApplication//BrowserWindow//input[@textContents='Login']
   ```

   The new locator displays in the **Selected Locator** text box.
7. Click **Test** to verify that the locator recognizes the correct control.
8. To copy the locator to the script, click **Paste**.
9. Click **Close**.

Starting an Application from Within a Script

Typically, when you record a script, a base state is automatically recorded that starts the application that you want to test. However, if your script tests multiple applications or you turn off the **Execute base state** functionality for any reason, you must start the test application from within your script.

1. Open the script in which you want to include the code to start the application.
2. Type `Imports System.Diagnostics` at the beginning of the script.
3. To define the application as a new process, type:
   ```
   Dim applicationProcess As New Process()
   ```
   where *applicationProcess* is the name of the application that you want to start.
   For example, to start Internet Explorer, type:
   ```
   Dim ieProcess As New Process()
   ```
4. Define the full path to the application by typing the following code:
   ```
   Dim psi As New ProcessStartInfo()
   psi.FileName = "applicationexecutable.exe"
   ```
   where *applicationexecutable.exe* is the name of the executable file that launches your test application.
   For example to start Internet Explorer, type:
   ```
   Dim psi As New ProcessStartInfo()
   psi.FileName = "C:\Program Files\Internet Explorer\iexplore.exe"
   ```
   Note: If the executable for the application that you want to test is in the same directory as Silk Test Workbench, you do not need to specify the full path.
5. To initially go to a certain Web page, set the `ProcessStartInfo.Arguments` property to the URL.
   For example, to start Internet Explorer and go to Google.com, add the following line after the `Filename` line:
   ```
   psi.Arguments = "http://www.google.com"
   ```
6. Specify the code to start the process.
   For example, to start `ieProcess` that was defined in step 1, type:
   ```
   ieProcess.StartInfo = psi
   ieProcess.Start()
   ```
7. Save the script.

The entire script to start Internet Explorer and go to Google.com looks like the following:

```
Imports System
Imports System.Diagnostics

Public Module Main

  Public Sub Main()
    Dim _desktop As Desktop = Agent.Desktop
    Dim ieProcess As New Process()
    Dim psi As New ProcessStartInfo()
    psi.FileName = "C:\Program Files\Internet Explorer
\iexplore.exe"
    psi.Arguments = "http://www.google.com"
    ieProcess.StartInfo = psi
    ieProcess.Start()

  End Sub
End Module
```

Recording an Object Map Item or a Locator Manually For a Script

Before you begin, ensure that the application that you want to test is running.

Manually capture an object map item or a locator using the **Identify Object** dialog box. Using the **Identify Object** dialog box enables you to identify objects that you want to test with ease and certainty.

1. Open the script in which you want to include the object and click the place within the script where you want to insert the object.
2. Choose **Tools** > **Identify Object**. The **Identify Object** dialog box opens.
3. Specify the **Selection mode**.
   - **Click** – Click the object to identify the locator.
   - **Hot Key** – Specify this mode to capture the object using the keystroke combination specified in the **Keystroke** list box. Typically, you choose this mode to capture objects, such as a menu or combobox, that only appear when clicked by the user. With this mode, you select the object and then press the hot key keystroke combination to capture the locator without dismissing the object.
4. Click **Start Identify**.

   If you want to capture an object map item or a locator in Google Chrome, Silk Test Workbench detects if the selected instance of Google Chrome is started with the appropriate automation parameters, and if not, Silk Test Workbench closes Google Chrome and restarts it with the automation parameters set.
5. *Optional:* To bring the application under test into the appropriate state before recording a locator, click **Stop Identify**. The actions that you perform in the application under test are no longer recorded. To continue with the recording of a locator, click **Start Identify**.
6. Position the mouse over the object that you want to record and perform one of the following steps:
   - If you use **Click** mode, click the object that you want to identify.
   - Press the keystroke combination to capture the object with the **Hot Key** mode.

     By default, the keystroke combination is `Ctrl+Shift`.

   Silk Test Workbench lists the related locator string in the **Selected Locator** text box.
7. To refine how locators display in the **Locator Details** table, perform any of the following actions:
   - **Hide Locator Details** – To hide the **Locator Details** table, click this link.
   - **Show object map names** – Check this check box to display object map item names in the **Locator** column. Object map item names associate a logical name (an alias) with a control or a window, rather than the control or window's locator. By default, object map item names are displayed. To use locators, uncheck this check box.
   - **Show full locators** – Check this check box to display the full locator name. To show only the attribute associated with the object, uncheck this check box.
   - **Show properties** – Check this check box to display any attributes and attribute values for the object selected in the **Locator Details** table. You can select attributes in this table to use in the locator identification. To hide the Properties subtree and display only locator details, uncheck this check box.
8. To test that the object that displays in the **Selected Locator** text box is the object that you want to use, click **Test**. Silk Test Workbench highlights the object that corresponds with the locator in the application that you want to test.
9. To replace the locator that you recorded, select the locator that you want to use in the **Locator Details** table. The new locator displays in the **Selected Locator** text box.
10. To copy the locator to the script, click **Paste**. Silk Test Workbench adds the locator to the script.
11. Modify the locator to compile correctly in the script and include the action that you want to perform.

    For instance, you can paste the following full locator into the script:

    ```
    /WPFWindow[@caption='Basic Controls'][1]//
    WPFRichTextBox[@automationId='editableTextBox']
    ```

    If the `With` statement already includes the `/WPFWindow` declaration, modify the script to use the following locator followed by a method:

    ```
    .WPFRichTextBox("@automationId='editableTextBox'").Select(1,10)
    ```

Recording Additional Actions Into an Existing Test

Once a test is created, you can open the test and record additional actions to any point in the test. This allows you to update an existing test with additional actions.

1. Open an existing test script.
2. Select the location in the test script into which you want to record additional actions.

   Note: Recorded actions are inserted after the selected location. The application under test (AUT) does not return to the base state. Instead, the AUT opens to the scope in which the preceding actions in the test script were recorded.
3. Click **Actions** > **Record**.

   Silk Test Workbench minimizes and the **Recording** window opens.
4. Record the additional actions that you want to perform against the AUT.

   During recording the Silk Test Workbench icon on the task bar flashes.For information about the actions available during recording, see *Actions Available During Recording*.
5. To stop recording, click **Stop** in the **Recording** window.

   You can also stop recording by pressing the stop recording key combination, which by default is **Alt +F10**, or by clicking the Silk Test Workbench task bar icon.

Opening an Existing Script

Open an existing script to play it back, review it, or make changes.

1. Choose **File** > **Open**. The **Asset Browser** opens.
2. Select **.NET Script** in the left pane to display the list of scripts.
3. In the right pane, double-click the asset that you want to open.

Editing a Script

After creating and saving a script, you can edit it at any time.

1. Open a script.
2. Edit the script just as you would in a text editor.

   You can select, delete, cut, copy, and paste text. The script window also has auto-completion and syntactical association technology, which helps to complete code with fewer errors.

   For example, when typing *Window* followed by a period, a list of all valid methods and properties for the Window object is displayed.

   Similarly, when typing a valid function followed by a space, comma or opening parenthesis, syntactical assistance is displayed.
3. Click **Save** on the toolbar or choose **File** > **Save** to enter a description and save the script.

Saving a Script

You can save a new or changed script. When you close an updated script, you are prompted to save any changes.

1. Choose **File** > **Save As**. The **Save As** dialog box opens.
2. *Optional*: Change the script name and enter a description.
3. Click **OK**.

Accessing the Definition of an Element in a Script

To access the definition of a method or a class in a VB .NET script, perform the following actions.

1. Open the script.
2. Right-click on the element for which you want to access the definition.

   You can also press **F12**.
3. Select **Go to Definition**.

Application Configuration

An application configuration defines how Silk Test Workbench connects to the application that you want to test. Silk Test Workbench automatically creates an application configuration when you create the base state. However, at times, you might need to modify, remove, or add an additional application configuration. For example, if you are testing an application that modifies a database and you use a database viewer tool to verify the database contents, you must add an additional application configuration for the database viewer tool.

- For a Windows application, an application configuration includes the following:
  - Executable pattern

    All processes that match this pattern are enabled for testing. For example, the executable pattern for Internet Explorer is `*\IEXPLORE.EXE`. All processes whose executable is named `IEXPLORE.EXE` and that are located in any arbitrary directory are enabled.
  - Command line pattern

    The command line pattern is an additional pattern that is used to constrain the process that is enabled for testing by matching parts of the command line arguments (the part after the executable name). An application configuration that contains a command line pattern enables only processes for testing that match both the executable pattern and the command line pattern. If no command-line pattern is defined, all processes with the specified executable pattern are enabled. Using the command line is especially useful for Java applications because most Java programs run by using `javaw.exe`. This means that when you create an application configuration for a typical Java application, the executable pattern, `*\javaw.exe` is used, which matches any Java process. Use the command line pattern in such cases to ensure that only the application that you want is enabled for testing. For example, if the command line of the application ends with **com.example.MyMainClass** you might want to use ***com.example.MyMainClass** as the command line pattern.
- For a web application in a desktop browser on the local machine, an application configuration includes only the browser type.

  Note: To start a browser with command line arguments or to specify a working directory or a specific executable for the browser, select the custom browser type. For additional information, see *Modifying the Base State*.
- For a web application in Apple Safari or in Microsoft Edge on a remote machine, an application configuration includes the following:
  - Browser type.
  - Connection string.
- For a web application in a mobile browser, an application configuration includes the following:
  - Browser type.
  - Connection string.
- For a native mobile application, an application configuration includes the following:

- Connection string.
- Simple application name. If multiple applications on the mobile device have the same name, the fully qualified name of the application is used.

Note: Do not add more than one browser application configuration when testing a web application with a defined base state.

Editing Application Configurations

An application configuration defines how Silk Test Workbench connects to the application that you want to test. Silk Test Workbench automatically creates an application configuration when you create the base state. However, at times, you might need to modify, remove, or add an additional application configuration. For example, if you are testing an application that modifies a database and you use a database viewer tool to verify the database contents, you must add an additional application configuration for the database viewer tool.

1. In the Silk Test Workbench menu, click **Tools** > **Edit Application Configurations**. If no project is selected, the **Select Project** dialog box appears.
2. Select a project from the list and click **Edit**. The **Edit Application Configurations** dialog box appears and lists the existing application configurations.
3. To add an additional application configuration, click **Add application configuration**.
   a) In the **Select Application** dialog box, select the tab that corresponds to the type of the application that you want to test.
   b) Select the application that you want to test.
   c) Click **OK**.

   Note: Do not add more than one browser application configuration when testing a web application with a defined base state.
4. To remove an application configuration, click **Remove**.
5. To edit an application configuration, click **Edit**.
6. Click **OK**.

Modifying an Application Configuration

An *application configuration* defines how Silk Test Workbench connects to and configures the application that you want to test.

1. Open the visual test or the VB .NET script.
2. For a visual test, click the **<<Start>>** step in the task pane.
3. In the **Properties** pane, navigate to the **Application Configurations** category.
4. Select the application configuration that you want to modify.
   - For a visual test, click into the application configuration that you want to modify and click **Edit application configuration**.
   - For a VB .NET script, double-click on the application configuration that you want to modify.

   The **Edit Application Configuration** dialog box opens.
5. To specify an executable, type the full path to the executable into the **Executable** field.

   Note: If you are testing a web application, and you want to specify an executable for the browser, select the custom browser type.

   For example, to start Mozilla Firefox, type `C:\Program Files (x86)\Mozilla Firefox\firefox.exe`.
6. If you are testing a desktop application and you want to use a command line pattern in combination with the executable file, type the command line pattern into the **Command Line Pattern** text box.

7. If you are testing a web application, type the address of the web application into the **Navigate to URL** text box.
8. *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
   For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.
9. *Optional:* Select an **Orientation** for the browser window.
10. *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.
11. If you are testing a mobile native application, specify the name of the mobile application that you want to test in the **Mobile app** text box.
12. Click **OK**.

Note: Do not add more than one browser application configuration when testing a web application with a defined base state.

Deleting an Application Configuration

Delete an application configuration from a visual test or from a VB .NET script to no longer test the corresponding application with this visual test or VB .NET script.

1. Open the visual test or the VB .NET script.
2. For a visual test, click the **<<Start>>** step in the **Task** pane.
3. In the **Properties** pane, navigate to the **Application Configurations** category.
4. Select the application configuration that you want to delete.
   - For a visual test, click into the application configuration.
   - For a VB .NET script, right-click the application configuration.
5. Click **Delete application configuration**.
6. Click **Yes** to confirm that you want to delete the application configuration.

Select Application Dialog Box

Use the **Select Application** dialog box to select the application that you want to test, to associate an application with an object map, or to add an application configuration to a test. Application types are listed in tabs on the dialog box. Select the tab for the application type you want to use.

Windows Lists all Microsoft Windows applications that are running on the system. Select an item from the list and click **OK**.

Use the **Hide processes without caption** check box to filter out applications that have no caption.

Web Lists all available browsers, including mobile browsers on any connected mobile devices and Apple Safari on a Mac. Specify the web page to open in the **Enter URL to navigate** text box. If an instance of the selected browser is already running, you can click **Use URL from running browser** to record against the URL currently displayed in the running browser instance. If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list. When selecting a browser size, you can also select an **Orientation** for the selected browser.

Note: Do not add more than one browser application configuration when testing a web application with a defined base state.

Mobile Lists all available mobile devices and all running Android emulators, including devices connected to a remote location. Select this tab to test mobile native applications. You can select to test the mobile application (app) that is currently running on the selected mobile device, or you can browse for or manually specify the name or file of the app that you want to test.

Editing Remote Locations

You can use the **Remote Locations** dialog box to add any browsers and mobile devices on a remote location to the set of applications that you can test.

1. Click **Tools** > **Edit Remote Locations**. The **Remote Locations** dialog box appears.
2. To add an additional remote location, perform the following actions:
   a) Click on the arrow to the right of **Add Location** to specify whether you want to add a remote location which is using the Silk Test Information Service, or Silk Central.

      Note: You can only configure one Silk Central as a remote location. If you have already configured the integration with Silk Central, Silk Central is listed in the remote locations list.
   b) Click **Add Location**. The **Add Location** dialog box appears.
   c) Type the URL of the remote location and the port through which Silk Test Workbench connects to the information service on the remote machine into the **Host** field.

      The default port is `22901`.
   d) *Optional:* Edit the name of the remote location in the **Name** field.
3. To edit an existing remote location, click **Edit**.
4. To remove a remote location, click **Remove**.
5. *Optional:* To reduce the amount of browsers and devices in the **Select Application** dialog, click **Do not show devices and browsers from this location**. The installed browsers and connected devices of the remote location will no longer be displayed in the **Select Application** dialog. By default, all installed browsers and connected devices of all remote locations are displayed in the **Select Application** dialog.
6. Click **OK**.

When you have added a remote location, the browsers that are installed on the remote location, including Apple Safari on a Mac, are available in the **Web** tab of the **Select Application** dialog box, and the mobile devices that are connected to the remote location are available in the **Mobile** tab of the **Select Application** dialog box.

User Account Control

You can use Silk Test Workbench with User Account Control (UAC). You can test an application with UAC enabled when the application has the same security level as Silk Test Workbench.

You need administrator permissions to install Silk Test Workbench on your system. When Silk Test Workbench is installed, you no longer need administrator permissions to work with Silk Test Workbench.

Note: If you want to use Silk Test Workbench with UAC enabled, we recommend you not to install Silk Test Workbench into the `Program Files` directory, because Silk Test Workbench requires writing permissions to the install directory. However, if you decide to install the Silk Test Workbench into the `Program Files` directory, you need administrator permissions to configure the database for Silk Test Workbench.

Disabling Specific Technology Domains

You can disable the loading of specific technology domains in a test, if the technology domains are not required for the test and are known to cause problems, for example by reducing the performance of the test.

Note: Disable technology domains only if you are certain that the technology domains are not required for the test.

1. To open the **Tech Domains** dialog box for a visual test:
   a) Select the **Start** step of the visual test.
   b) Open the **Properties** tab.
   c) Click into the **Tech Domains** field.
   d) Click **Open**.
2. To open the **Tech Domains** dialog box for a VB .NET script:
   a) Open the **Properties** tab.
   b) In the properties tree, right-click on the application configuration for which you want to disable the technology domains.
   c) Click **Select Tech Domains**.
3. In the **Tech Domains** dialog box, uncheck the **Automatically load Tech Domains as necessary** check box.
4. In the **Tech Domain** list, uncheck the technology domains that you no longer want to load for the test.
5. Click **OK**.

In future executions of the test, only the enabled technology domains will be loaded.

Application Configuration Errors

When the program cannot attach to an application, the following error message opens:
Failed to attach to application <Application Name>. For additional information, refer to the Help.

In this case, one or more of the issues listed in the following table may have caused the failure:

Issue	Reason	Solution
Time out	• The system is too slow. • The size of the memory of the system is too small.	Use a faster system or try to reduce the memory usage on your current system.
User Account Control (UAC) fails	You have no administrator rights on the system.	Log in with a user account that has administrator rights.
Command-line pattern	The command-line pattern is too specific. This issue occurs especially for Java. The replay may not work as intended.	Remove ambiguous commands from the pattern.
• The **Select Browser** dialog box does not display when running a test against a Web application. • Multiple browser instances are started when running a test against a Web application. • When running a test against a Web application with a browser instance open, Silk Test Workbench might stop working.	A base state and multiple browser application configurations are defined for the test case.	Remove all browser application configurations except one from the test case.

Issue	Reason	Solution
Playback error when running a test	No application configuration is defined for the test. The following exception might be displayed: `No application configuration present.`	• When running a keyword-driven test, ensure that a keyword which executes the base state is included in the test. • Ensure that an application configuration is configured for the current project.

Troubleshooting Application Configurations

Why is my application not displayed in the Select Application dialog box

- Uncheck the **Hide processes without caption** check box. This check box is checked by default and prevents applications without a caption from being displayed in the dialog box.
- Run Silk Test Workbench with elevated privileges.
  1. Close Silk Test Workbench.
  2. Stop the Open Agent.
  3. Run Silk Test Workbench as an administrator.
- Use the **Task Manager** to check if the application is running under a different user account.
- Ensure that the application is not started with the `runas` command or a similar command.

Environment Variables in Application Configurations

The following environment variables might be used in application configurations.

Environment Variable	Description
TMP or TEMP	The Windows temporary directory.
USERPROFILE	The user profile directory (C:\Documents and Settings *myUserName*).
APPDATA	The application data directory.
ProgramFiles	The Program Files directory.
ProgramFiles(x86)	The Program Files directory for 32-bit applications, which is only available for 64-bit environments.

Base State

An application's *base state* is the known, stable state that you expect the application to be in before each test begins execution. This state may be the state of an application when it is first started. The base state is automatically generated when you select the application that you want to record.

When you record a visual test or script, Silk Test Workbench automatically creates a base state. You can turn the base state on and off to accommodate your test needs. For instance, in certain situations you might want to launch the application window manually. In those instances, turn off the base state. The base state information is part of the application configuration.

Base states are important because they ensure the integrity of your tests. By guaranteeing that each test can start from a stable base state, you can be assured that an error in one test does not cause subsequent tests to fail.

Silk Test Workbench ensures that your application is at its base state before a visual test or script runs.

When an error occurs, Silk Test Workbench stops execution of the test.

Turning the Base State On and Off

When you record a visual test or script, Silk Test Workbench automatically creates a base state. You can turn the base state on and off to accommodate your test needs. For instance, in certain situations you might want to launch the application window manually. In those instances, turn off the base state.

1. In a visual test, perform the following steps:
   a) In the **Test Steps** pane, click the **<<Start>>** step.
   b) In the **Properties** pane, navigate to the **Application Configurations** category.
   c) Click into the application configuration field and click **Edit Application Configuration**. The **Edit Application Configuration** dialog box appears.
   d) Perform one of the following steps:
      - To use the base state, check the **Execute Base State** check box.
      - To turn off the base state, uncheck the **Execute Base State** check box.
   e) Click **OK**.
2. In a script, perform the following steps:
   a) In the **Properties** pane, right-click the application configuration that you want to modify and choose **Edit *Application Executable Name***.
   For example, if you are testing a web application with Internet Explorer, choose **Edit iexplore.exe** from the menu.
   The **Edit Application Configuration** dialog box appears.
   b) Perform one of the following steps:
      - To use the base state, check the **Execute Base State** check box.
      - To turn off the base state, uncheck the **Execute Base State** check box.
   c) Click **OK**.

Running the Base State

Before starting to record a test against an application, you can execute the base state to bring all applications, against which you want to record, to the appropriate state for recording.

When you execute the base state, all application configurations in the active visual test or .NET script, which have **Execute BaseState** set, are executed.

To run the base state:

Click **Actions** > **Execute BaseState**.

Depending on the type of the application, Silk Test Workbench performs the following actions:

- Executes the application configurations of all applications, for which an application configuration is defined in the current project.
- For web applications, Silk Test Workbench opens the web application in the specified browser and to the specified URL.
- For mobile web applications, Silk Test Workbench opens the specified browser on the specified mobile device or Emulator to the specified URL.
- For mobile native applications, Silk Test Workbench opens the specified app on the specified mobile device or Emulator. If the specified app is not installed on the specified mobile device or Emulator, Silk Test Workbench installs and then opens the app.

Modifying the Base State from the User Interface

You can edit the base state from the user interface to specify how Silk Test Workbench starts an application under test (AUT) during recording and replay. You can specify the executable location of the AUT, the working directory, the URL and the connection string for a web application, and so on. For example, if you want to execute tests on a production web site, which have already been executed on a staging web site, you can simply change the URL in the base state and the tests are executed against the new web site.

Note: To specify how Silk Test Workbench starts an application under test (AUT) only for specific tests during replay, edit the base state in the script that contains the tests. For additional information, see *Modifying the Base State in a Script*.

To edit the base state through the user interface:

1. Open the visual test or the VB .NET script.
2. For a visual test, click the **<<Start>>** step in the task pane.
3. In the **Properties** pane, navigate to the **Application Configurations** category.
4. Select the application configuration that you want to modify.
   - For a visual test, click into the application configuration that you want to modify and click **Edit application configuration**.
   - For a VB .NET script, double-click on the application configuration that you want to modify.

   The **Edit Application Configuration** dialog box opens.
5. Check the **Execute Base State** check box to use the base state with the application configuration.
6. To specify an executable, type the full path to the executable into the **Executable** field.

   Note: If you are testing a web application, and you want to specify an executable for the browser, select the custom browser type.

   For example, to start Mozilla Firefox, type `C:\Program Files (x86)\Mozilla Firefox\firefox.exe`.
7. To specify command line arguments, type the arguments into the **Command Line Arguments** field.

   Note: If you are testing a web application, and you want to start a browser with command line arguments, select the custom browser type.

   For example, to start Mozilla Firefox with the profile *myProfile*, type `-p myProfile`.
8. If you are testing an application which depends on a supplemental directory, specify the path to the directory in the **Working directory** filed.

   For example, if you use a batch file to start a Java application, the batch file may reference a JAR file that relies on a relative path. In this case, specify a working directory to reconcile the relative path.
9. If you are testing a desktop application, specify the main window of the application in the **Locator** field. For example, the locator might look like `/Shell[@caption='Swt Test Application']`.
10. If you are testing a desktop application and you want to use an executable pattern, type the executable name and file path of the desktop application that you want to test into the **Executable Pattern** text box. For example, you might type `*\calc.exe` to specify the Calculator.
11. If you are testing a desktop application and you want to use a command line pattern in combination with the executable file, type the command line pattern into the **Command Line Pattern** text box.
12. If you are testing a web application, type the address of the web application into the **Navigate to URL** text box.
13. *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
    For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.

14. *Optional:* Select an **Orientation** for the browser window.
15. *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.
16. If you are testing a web application or a mobile native application on a remote location, for example on a mobile device that is connected to a Mac, and you want to edit the remote location, click **Change** to open the **Select Application** dialog box and then click **Edit Remote Locations**.
17. If you are testing a mobile application or a web application on Apple Safari, type the connection string into the **Connection String** text box.

    For additional information, see *Connection String*.
18. To edit the capabilities for a WebDriver-based browser, you can use the **Connection String** text box. For example, to start Google Chrome with a maximized browser window, type the following into the **Connection String** text box:

    ```
    "chromeOptions="
    + "{"
    + " \"args\":[ \"--start-maximized\"]"
    + "};"
    ```

    For additional information, see *Setting Options and Capabilities for Web-Driver Based Browsers*.
19. If you are testing a mobile native application, specify the name of the mobile application that you want to test in the **Mobile app** text box.
20. Modify the **Timeout** to specify the number of milliseconds that Silk Test Workbench waits for the application to be ready and running when the base state is executed. If the application is not ready and running when the time expires, the process times out.
21. Click **OK**.
22. If the application under test usually takes a long time to start, increase the application ready timeout in the replay options.

Executing the base state starts the application if it is not already running. If the application is already running, Silk Test Workbench does not start another instance of the application.

If your test includes multiple application configurations and you are modifying an application or Web page other than the object associated with the base state, you can turn off the base state. This indicates that the base state will not be used for recording or replaying the modifications. Therefore, you must record the steps to launch the application or Web page within your test. For instance, if you want to test a Web page, start Internet Explorer within your test.

Note: Do not add more than one browser application configuration when testing a web application with a defined base state.

Modifying the Base State in a Script

You can edit the base state in a script to specify how Silk Test Workbench starts an application under test (AUT) during replay. You can specify the executable location of the AUT, the working directory, the URL and the connection string for a web application, and so on. For example, if you want to execute tests on a production web site, which have already been executed on a staging web site, you can simply change the URL in the base state and the tests are executed against the new web site.

Note: To specify how Silk Test Workbench starts an application under test (AUT) during recording and replay, edit the base state from the user interface. For additional information, see *Modifying the Base State*.

To edit the base state in a script:

1. Open the script.
2. Change the `baseState` method.

You can add your code between creating the base state and executing it:

```
' VB code
' Go to web page 'demo.borland.com/InsuranceWebExtJS'
BrowserBaseState baseState = new BrowserBaseState()
' <-- Insert your changes here!
baseState.Execute()
```

3. Use the following code to specify the executable name and file path of the application that you want to test:

   ```
   ' VB code
   baseState.Executable=executable
   ```

   For example, to specify the Calculator, type the following:

   ```
   ' VB code
   baseState.Executable = "C:\\Windows\\SysWOW64\\calc.exe"
   ```

   To specify Mozilla Firefox, type the following:

   ```
   ' VB code
   baseState.Executable = "C:\\Program Files (x86)\\Mozilla Firefox\
   \firefox.exe"
   ```

4. Use the following code to specify command line arguments:

   ```
   ' VB code
   baseState.CommandLineArguments = commandLineArguments
   ```

   For example, to start Mozilla Firefox with the profile *myProfile*, type the following:

   ```
   ' VB code
   baseState.CommandLineArguments = "-p myProfile"
   ```

5. You can use the following code to specify a different working directory:

   ```
   ' VB code
   baseState.WorkingDirectory = workingDirectory
   ```

6. If you want to use an executable pattern, use the following code:

   ```
   ' VB code
   baseState.ExecutablePattern = executablePattern
   ```

   For example, if you want to specify an executable pattern for the Calculator, type:

   ```
   ' VB code
   baseState.ExecutablePattern = "*\\calc.exe"
   ```

7. If you want to use a command line pattern in combination with the executable file, use the following code:

   ```
   ' VB code
   baseState.CommandLinePattern = commandLinePattern
   ```

   For example, if the command line of the application ends with **com.example.MyMainClass** you might want to use ***com.example.MyMainClass** as the command line pattern:

   ```
   ' VB code
   baseState.CommandLinePattern = "*com.example.MyMainClass"
   ```

8. If you are testing a web application or a mobile web application, and you have not set an application configuration for the current project, specify one of the installed browsers or mobile browsers.
   For example, to specify Google Chrome, type the following:

   ```
   ' VB code
   baseState.BrowserType = BrowserType.GoogleChrome
   ```

9. If you are testing a web application or a mobile web application, and you have not set an application configuration for the current project, specify the address of the web application that you want to test:

   ```
   ' VB code
   baseState.Url = url
   ```

For example, type the following:

```
' VB code
baseState.Url = "demo.borland.com/InsuranceWebExtJS/"
```

10. If you want to test a web application on a desktop browser, you can specify the height and width of the browser window:

```
' VB code
baseState.ViewportHeight = viewportHeight
baseState.ViewportWidth = viewportWidth
```

11. If you want to test a web application or a mobile native application on a remote location, specify the connection string:

```
' VB code
New MobileBaseState(connectionString)
```

For information on the connection string, see *Connection String for a Remote Desktop Browser* or *Connection String for a Mobile Device*.

12. To edit the capabilities for Mozilla Firefox or Google Chrome , you can also use the connection string. For example, to set the download folder for Mozilla Firefox, type the following:

```
' VB code
baseState.ConnectionString = "moz:firefoxOptions= {\"prefs\":
{ \"browser.download.dir\":\"C:/Download/\"}};"
```

For additional information, see *Setting Options and Capabilities for Web-Driver Based Browsers*.

Recording Overview

When you record a new visual test or script, the **Select Application** dialog box opens. Double-click the application that you want to test. Silk Test Workbench automatically creates an application configuration and base state to identify the test application. An *application configuration* defines how Silk Test Workbench connects to and configures the application that you want to test. An application's *base state* is the known, stable state that you expect the application to be in before each test begins execution.

If you open the **Identify Object** dialog box before you record a visual test or script, the **Select Application** dialog box opens also.

After you select the application that you want to test, perform one of the following steps:

- Record a visual test
- Record a script
- Select the object that you want to identify

You can modify the application configuration and turn the base state off if necessary.

Highlighting Objects During Recording

During recording, the active object in the AUT is highlighted by a green rectangle. As soon as a new object becomes active this new object is highlighted. If the same object remains active for more than 0.5 seconds a tool-tip will be displayed that displays the class name of the active object and also the current position of the mouse relative to the active object. This tool-tip will no longer be displayed when a new object becomes active, the user presses the mouse, or automatically after 2 seconds.

Characters Excluded from Recording and Replaying

The following characters are ignored by Silk Test during recording and replay:

Characters	Control
...	`MenuItem`
tab	`MenuItem`
&	All controls. The ampersand (&) is used as an accelerator and therefore not recorded.

Actions Available During Recording

During recording, you can perform the following actions in the **Recording** window:

Action	Steps
Pause recording.	Click **Pause** to bring the AUT into a specific state without recording the actions, and then click **Record** to resume recording.
Change the sequence of the recorded actions.	To change the sequence of the recorded actions in the **Recording** window, select the actions that you want to move and drag them to the new location.
Select multiple actions.	To select multiple actions, press `Ctrl` and click on the actions or press **Shift** and click on the first and the last action that you want to select.
Replay recorded actions.	To replay recorded actions from the **Recording** window, select the actions and click **Play**. To select all recorded actions, click on **Recorded Actions** and then click **Play**.
Remove a recorded action.	To remove a falsely recorded action from the **Recording** window, hover the mouse cursor over the action and click **Delete**.
Verify an image or a property of a control.	Move the mouse cursor over the object that you want to verify and press **Ctrl +Alt**.
Change the object map entry	If you are recording against a web application or a mobile web app and if the automatically generated object map entry for a recorded object is difficult to read or contains special characters, you might want to change the object map entry to something more readable. You can do this during recording by right-clicking on the object and then expanding the **Object identification** area of the **Choose Action** dialog. Then you can edit the object map entry in the **Object Map ID** field. For example, the automatically generated object map entry for an image in our demo application is *http demo borland*. If you look at the object map, it is be difficult to understand what object this entry refers to. Changing the object map entry to something like *InsuranceWebHomePageBanner* would possibly provide more context. This functionality is available for all supported desktop and mobile browsers except Internet Explorer.
Select a different locator	If you are recording against a web application or a mobile web app and the automatically generated locator for a recorded object does not meet your requirements, you can click on the arrow in the **Locator** field and let Silk Test Workbench generate alternative locator suggestions for you. All suggested locators uniquely identify the object. This functionality is available for all supported desktop and mobile browsers except Internet Explorer.

Enhancing Tests

Describes different methods for enhancing a visual test or script.

ActiveData

Visual tests and scripts usually contain constant or literal data values, which automatically enter data into an application's open fields or multiple-choice controls. When testing such applications, the visual tests and scripts use the same data field names and data values for each transaction.

To repeatedly play back visual tests and scripts using data other than the literal data, you must modify the visual tests or scripts to use different data values.

ActiveData testing enables you to leverage existing data in external files as input for powerful, comprehensive application testing solutions. With ActiveData, you can perform multiple transactions against test applications using a different set of data for each transaction without writing complicated code or compromising existing data.

You can write new data to existing data files without having to edit those data files manually.

Using ActiveData, testers can input valid or non-valid data into the test application and verify the results. This means fewer visual tests or scripts are needed to carry out a series of test conditions and less maintenance overhead.

Note: If an error occurs while opening an ActiveData file, use error handling to continue the test execution.

Visual Tests

With visual tests, you can use ActiveData where you use local variables. Substitute input with data from an ActiveData file, or create repetition logic to cycle through a series of input steps and substitute recorded data with data from an ActiveData file. You can also write any data created in a visual test, such as the contents of an expression, to an ActiveData file and use it for testing.

Select ActiveData files to use with a visual test in the ActiveData property of the visual test's `<<Start>>` step, or select a step and choose **Insert** > **ActiveData**.

Scripts

With scripts, you can insert ActiveData where you use local variables. However, unlike with visual tests, no wizard is available. You must manually insert the data.

Keyword-driven tests

With keyword-driven tests, you cannot simply use active data for the entire test. You can however use active data within the visual tests or .NET scripts that implement the individual keywords. To use active data within such a keyword, refer to the topics that describe using active data with the implementing type of the keyword.

Benefits of ActiveData Testing

Much of application testing involves data input into similar controls in an application's forms, then processing the data and reporting results to ensure the application works as expected. When automated through Silk Test Workbench, testers can validate the reliability of applications often using only the set of data that was input during recording.

With the ActiveData testing feature, testers can leverage existing and new data in their own data banks such as customer records for data input to create powerful automated application testing solutions. Automated tests using ActiveData can run multiple transactions through an application, using a different set of existing data for each transaction.

Data used for input can be managed separately or within Silk Test Workbench. Making changes to data between playback runs provides even greater and more efficient testing flexibility.

ActiveData Assets

When you create an ActiveData asset, you select the external file to use with the asset. You also determine what data in the external file (known as the ActiveData file) to use, whether the ActiveData file contains column headings, and the read/write permissions for the file associated with the ActiveData asset. Once an ActiveData asset is created, it can be used in any visual test or script and modified at any time.

Note: If Silk Test Workbench attempts to access an open ActiveData file a runtime error occurs. Use error handling to continue the test execution in this situation.

Data files can be either prepared in advance or created and edited within Silk Test Workbench. Create, modify, and maintain ActiveData assets using the **Asset Browser**.

Creating an ActiveData Test Asset

When creating an ActiveData asset, you either create a new file to contain the data used by the asset, or select an existing file that contains data to be used. To use an existing file, make sure the file exists and is available prior to creating the asset.

1. Choose **View** > **Asset Browser**.
2. Under **Asset Types**, right-click **ActiveData** and choose **New ActiveData**. The **ActiveData** window opens.
3. Type a descriptive name for the asset in the **Name** text box.

   ActiveData asset names must follow Silk Test Workbench asset naming conventions.

   Tip: Giving the ActiveData asset the same name as the ActiveData file used for the asset helps to quickly identify the asset for test use.
4. Click **Browse** to select an existing file to use as the ActiveData file, or click **New** to create a new file to use as the ActiveData file. The ActiveData file path and file name display in the **File** text box.
5. *Optional*: Type a description for the asset in the **Description** text box.

   Use the description to indicate the type of file used as the data file, the type of testing being performed by the ActiveData test for which the asset is being created, or the application being tested with the ActiveData test. The description displays in the **Asset Browser** for any ActiveData assets that include a description.

   Note: If an empty data file is required, click **Save** and **Close** to save any changes and close the **ActiveData** setup window.
6. Continue creating the ActiveData asset by setting read options for the data file being used by the ActiveData asset.
7. Continue creating the ActiveData asset by editing the data in the ActiveData file.
8. Click **Save** to save any changes, or click **Save and Close** to save any changes and close the **ActiveData** window.

The new asset displays in the ActiveData asset listing in the **Asset Browser**. You can use it in ActiveData testing.

Preparing Test Files for ActiveData Testing

Silk Test Workbench can use the following file types for ActiveData testing:

- Microsoft Excel® Spreadsheet (.xls) files.

  Note: If Microsoft Excel® is not installed, you can use .xls files in read-only mode.
- Microsoft Excel® Spreadsheet (.xlsx) files.

  Note: To use a .xlsx file with Silk Test Workbench, install Microsoft Excel® 2007 or later. If Microsoft Excel® 2007 or later is not installed, you can install *2007 Office System Driver: Data Connectivity Components* to use .xlsx files in read-only mode.
- Plain text files (.txt)
- Comma separated value (.csv) files

Comma separated value and plain text files can only contain one set of data, but Excel spreadsheet files can contain multiple sheets. Each sheet contains its own table of values and is treated as a separate entity in ActiveData testing.

When creating a new test data asset using a comma separated value or plain text file, only the file needs to be selected. When creating a new test data asset using an Excel spreadsheet file, the sheet to use must also be chosen.

Tips for Preparing Files for ActiveData Testing

- When possible, data in data files should be grouped into the same data categories as they exist in the application. For example, if the data file contains First and Last Names that will be used as ActiveData, the application being tested should have separate fields for a First Name and a Last Name. If the application has separate fields for a First and Last Name, but the data file has the first and last name combined into one data field, there is no one-to-one correspondence between the test data and what is required to test the application. Using the combined data from the ActiveData file in testing a first name or last name field in the test application may produce undesirable results.
- Files used for ActiveData testing must be created before you use them in a visual test or script.
- For object name test data, the name entries in the data file should not contain parentheses or quotes.

  For example, if performing the same test on multiple edit boxes on one form, and only the index property is used to make each edit box unique, the name for each EditBox might be represented as follows:

  ```
  TextField("[1]")
  TextField("[2]")
  TextField("[3]")
  TextField("[4]")
  ```

  Each row of object name test data in the ActiveData file could contain a separate index to represent the edit boxes on the form. However, the data in each row of the ActiveData file should not contain the parentheses or the quotes:

  ```
  Index=1
  Index=2
  Index=3
  Index=4
  ```
- The first row of an ActiveData file can either be the first row of data, or a column heading. When setting up an ActiveData test, the row type for the first row of the ActiveData file must be properly set in the ActiveData asset.
- If a user opens an ActiveData asset as read-only, then only read permission is given. If an ActiveData asset is opened in read-write mode, both read and write permissions are given, even if the asset is never written to. This is done to ensure proper playback if a test is modified to write to an ActiveData file at a later time. An ActiveData asset opened with read-write permission has an exclusive file lock, preventing other programs from opening the ActiveData file associated with the asset.

Creating a New Data File for ActiveData Testing

ActiveData assets require a data file to be associated and used with the asset. Create the file with Silk Test Workbench or use tools such as text editors or Microsoft Excel® to create the file.

While creating an ActiveData asset, you can also create a new ActiveData file within Silk Test Workbench that contains the ActiveData you want to associate with the asset.

1. Choose **View** > **Asset Browser**.
2. Under **Asset Types**, right-click **ActiveData** and choose **New ActiveData**. The **ActiveData** window opens.
3. Type a descriptive name for the asset in the **Name** text box.

   ActiveData asset names must follow Silk Test Workbench asset naming conventions.

   Tip: Giving the ActiveData asset the same name as the ActiveData file used for the asset helps to quickly identify the asset for test use.
4. Click **New** to create a new file to use as the ActiveData file. The **New ActiveData** dialog box opens.
5. In the **Name** text box, type a descriptive name for the new file.

   It is a good practice to give the file a name that either indicates what it contains or the type of application it will be used to test.
6. Select the file type for the new ActiveData file.

   For file type information, see *Creating an ActiveData Test Asset*.
7. Click **Browse** and use the **Browse for Folder** dialog box to select the location to store the new ActiveData file. Then, click **OK** to return to the **New ActiveData** dialog box. The location for the new ActiveData file displays in the **Location** text box.
8. Click **OK** to create the new ActiveData file.
9. Continue creating the ActiveData asset.

Setting Read Options for ActiveData Test Files

Set the read/write options for an ActiveData file using the **Options** tab of the **ActiveData** setup window. Set whether to treat the first row in the ActiveData file as a column header, and set the sheet in a spreadsheet data file to use for ActiveData testing using this tab as well. The procedure for setting these options differs slightly for a spreadsheet data file as opposed to text (.txt) or comma separated (.csv) data files.

1. In the **Options** tab of the **ActiveData** window, check the **Use first row as header** check box to allow Silk Test Workbench to treat the first row of data in the file as column headers.

   Checking this check box enables you to map data to records held in rows for each column in the data file when creating an active data test. If the **Use first row as header** check box is not checked, data in the first row of the file is interpreted as actual ActiveData items.
2. Check the **Force read only** check box to force Silk Test Workbench to treat the ActiveData file as a read only file, regardless of whether or not the file is actually set to be read only.

   Selecting this option does not change the read/write setting for the ActiveData file.

   Checking the **Force read only** check box also disables the **Insert** and **Delete** buttons for rows and columns in the **Details** tab of the **ActiveData** window, so changes cannot be made to the ActiveData file from within Silk Test Workbench.
3. For spreadsheets, in the **Sheet** list, select the sheet within the Excel spreadsheet that contains the data to be used for the ActiveData asset.
4. For comma separated value and text files, select an option to indicate the character used in the ActiveData file to separate data items.

If using a custom character, select the **Other** option and type the custom character in the **Other** text box.

5. Continue creating the ActiveData asset.

Editing ActiveData Files for ActiveData Testing

Edit data in an ActiveData file used for an ActiveData asset using the **Details** tab of the **ActiveData** window. Functionality in the **Details** tab is disabled until an ActiveData file is named in the **General** tab. If the **Force read only** check box is not checked, columns and rows can be added to and removed from the data file, and data can be copied and pasted to and from the list.

Tip: You can also edit ActiveData files in the file's native application. Any changes made to an ActiveData file using its native application appear in the file the next time it is opened in Silk Test Workbench.

1. Update the contents of the ActiveData file as required.
   a) Click the appropriate button to insert a row or column. Rows and columns are automatically inserted after the last existing row or column.
   b) Select a cell in a row or column, then click the appropriate **Delete** button to delete the row or column, then click **Yes** in the message box to confirm the deletion. All data for all cells in the row or column is deleted as well.
   c) Double-click in a cell to enter or edit data in that cell. Data can be typed or copied from an external source and pasted into a cell.
   d) Select a cell in a column, then click **Rename** to change the name of the header for that column.
   e) Click **Save Data** at any time to save any updates that have been made. Clicking **Save Data** saves changes back to the original ActiveData file. If only changing data for a particular test run, first save a copy of the original ActiveData file.
2. Continue creating the ActiveData asset.

   You can update data from an ActiveData file used in ActiveData testing during playback. For a visual test, updates are automatically saved back to the ActiveData file, or you can create steps to control how data updates are saved back to the ActiveData file. For more information, see *Modifying Advanced Options*. For scripts, ActiveData files can be edited during playback using .NET commands.

   Tip: When data in the ActiveData file is updated and saved using the **Details** tab, Silk Test Workbench updates the version number of the corresponding ActiveData asset. However, when updating data in the ActiveData file using an external tool such as Microsoft Excel, the asset version number is not updated.

   Tip: Add data to always contain quotes or trailing spaces as required for ActiveData testing.

Determining Data Use for ActiveData

When using data in an ActiveData file for ActiveData testing, you can specify the rows of data in the file that are used and the order in which they are used. The **Start row**, **End row**, and **Random count** values determine the rows used and the order in which they are used.

Random count can take different values that determine if and how records from the ActiveData file associated with the asset are used in the ActiveData test.

For visual tests, set the **Start row**, **End row** and **Random count** values in the **Define the ActiveData asset to use** page of the **Test Logic (Repetition) Designer**. Once set, you can update the values in the **Start row**, **End row** and **Random count** properties for the *Repeat using ActiveData* step. For VB .NET test scripts, set and update the **Start row**, **End row** and **Random count** values using the `StartRow`, `EndRow`, and `RandomCount` parameters of the `LoadActiveData` method.

The following table shows the result of using different **Start row** and **End row** values with the acceptable values for **Random count**.

Start Row Value	End Row Value	RandomCount Value (Visual Test)	RandomCount Value (VB .NET Script)	Result
1	-1 [b]	Get all rows in sequence	0 or False (Get all rows in sequence)	All rows load in index order.
1	5	Get all rows in sequence	0 or False (Get all rows in sequence)	Rows 1, 2, 3, 4, and 5 load in order.
5	-1 [b]	Get all rows in sequence	0 or False (Get all rows in sequence)	Row 5 through the last row in the ActiveData file load in order.
5	1	Get all rows in sequence	0 or False (Get all rows in sequence)	Rows 5, 4, 3, 2, and 1 load.
-1 [a]	1	Get all rows in sequence	0 or False (Get all rows in sequence)	All rows load in reverse order
-1 [a]	5	Get all rows in sequence	0 or False (Get all rows in sequence)	Loads the last row in the ActiveData file through row 5 in reverse order.
-1 [a]	-1 [b]	Get all rows in sequence	0 or False (Get all rows in sequence)	Loads the last row in the ActiveData file.
1	1	Get all rows in sequence	0 or False (Get all rows in sequence)	Loads row 1.
1	-1 [b]	Get all rows in random order	-1 or True (Get all rows in random order)	Loads all rows in random order.
1	5	Get all rows in random order	-1 or True (Get all rows in random order)	Rows 1 through 5 load in random order.
5	-1 [b]	Get all rows in random order	-1 or True (Get all rows in random order)	Rows 5 through the last row in the ActiveData file load in random order.
5	1	Get all rows in random order	-1 or True (Get all rows in random order)	Loads rows 5 through 1 in random order.
-1 [a]	1	Get all rows in random order	-1 or True (Get all rows in random order)	Loads all rows in random order.
-1 [a]	5	Get all rows in random order	-1 or True (Get all rows in random order)	Loads rows 5 through the last row in the ActiveData file in random order.
1	-1 [b]	5	5	Loads five random rows from the ActiveData file.
1	10	5	5	Loads five rows from the first through the tenth row in the ActiveData file in random order.
10	-1 [b]	5	5	Loads five rows from the tenth row through the last row in the ActiveData file in random order.
1	5	10	10	Generates an out-of-bounds run-time error.

[a] For visual test, checking the **Start at last row containing data** check box for the **Start row** value is the equivalent.

[b] For visual test, checking the **End at last row containing data** check box for the **End row** value is the equivalent.

If any value used for the **StartRow**, **EndRow**, or **RandomCount** exceeds the number of rows in the ActiveData file, an out-of bounds run-time error generates.

A run-time error occurs if `LoadActiveData()` attempts to load an ActiveData file that is opened in another application.

Using ActiveData in Visual Tests

To use ActiveData in a visual test, any data files to be used must be created. Each data file to be used must be associated with an ActiveData asset.

Test data is stored in and retrieved from outside the Silk Test Workbench database, using external files. Test data can reside in Excel Spreadsheet files (.xls and .xlsx), text files (.txt), or comma separated value (.csv) files. Storing test data in external files lets you reuse test data in more than one ActiveData asset and in any visual test or script.

Note: If Microsoft Excel® is not installed, all .xls files are in read-only mode. To use a .xlsx file with Silk Test Workbench, install Microsoft Excel® 2007 or later. If Microsoft Excel® 2007 or later is not installed, you can install *2007 Office System Driver: Data Connectivity Components* to use .xlsx files in read-only mode.

Note: If an error occurs while opening an ActiveData file, use error handling to continue the test execution.

Visual tests must specify which ActiveData assets are available for use in the visual test. Properties for each step in a visual test that are to use ActiveData must be modified to point to the correct data. This is known as *mapping*.

Associating an ActiveData Asset with a Visual Test

Before using an ActiveData asset in a visual test, you must associate it with the visual test. Any number of ActiveData assets can be associated with a visual test.

Note: ActiveData assets may be inserted/created while any test step of a visual test is selected. However, the properties are displayed in the **ActiveData** property of the `<<Start>>` step.

1. Open the visual test in which you want to use ActiveData.
2. To create a new ActiveData asset, choose **Insert** > **ActiveData** > **New**.
3. To browse for the ActiveData asset, choose **Insert** > **ActiveData** > **Associate Existing**.

   The **Browse for ActiveData** dialog box opens.

   a) Select whether the asset list should display all assets or only the ones created by you.
   b) Select the asset from the list.

      Note: Click any column header (**Name**, **Project**, **Description**, or **Modified Date**) to alphabetically sort the list of ActiveData assets by column header type.

   c) Click **OK**.

      Tip: Silk Test Workbench displays a message if the ActiveData file being associated does not contain rows or columns of data. Click **Yes** in the message box to associate the ActiveData asset with the visual test, or **No** to cancel.
4. Repeat the preceding steps to associate ActiveData files from other ActiveData assets.

   Each time an ActiveData asset is associated with the visual test, an additional property named Item [*n*] appears under the ActiveData property, where [*n*] is a sequential number in the total number of ActiveData assets associated with the visual test. The name of the selected ActiveData asset displays as the value of this property. The **ActiveData** property is in the **General** property category.
5. Edit ActiveData associations for a visual test by clicking the `<<Start>>` step.

   Properties for the step display in the **Properties** pane. Sort properties by category or alphabetical order. To edit an association, click the appropriate **Item** property in the **Properties** pane.

   - Click the **Browse for ActiveData** toolbar button in the value area to replace the associated ActiveData asset using the **Browse for ActiveData** dialog box.

- Click the **Delete** toolbar button to remove the association of the ActiveData asset with the visual test.

Creating Repetition Logic for ActiveData in a Visual Test

Creating repetition logic for ActiveData allows a visual test to cycle through data in an ActiveData file and substitute the literal data in steps with the data in the ActiveData file.

When creating repetition logic for ActiveData in a visual test, you essentially define a sequential block of test steps that are used for ActiveData testing. Steps in the defined block that substitute literal data with data in an ActiveData file repeat during playback. The number of times the steps repeat is defined in the repetition logic, which can be all the rows in the ActiveData file or a subset of the rows.

Creating repetition logic for ActiveData in a visual test does not mean the visual test now uses ActiveData. This part of the process sets up steps in a test, but data in the steps themselves must be mapped to data in an ActiveData file.

1. In the visual test, select the step that precedes the first step containing data to be substituted.
2. Click **Insert** > **Test Logic** > **Repetition**. The **Test Logic Designer** wizard opens.
3. If the **General** option to display wizard welcome screens is set to **Yes**, the **Welcome** page opens. Click **Next** to display the **Select a Logic Type (Repetition)** page.

   Tip: If the **General** option to display wizard welcome screens is set to **No**, the **Select a Logic Type (Repetition)** page appears as the first page of the **Test Logic Designer** wizard.
4. In the **Select a Logic Type (Repetition)** page, select the **Repeating a sequence of steps using data from an ActiveData file** option and click **Next**. The **Define the ActiveData asset to use** page appears.
5. Select the ActiveData asset that contains the data file to be used.
6. *Optional:* From the **Sheet name** list, select the sheet to be used from the data file.

   By default, the sheet that is specified in the ActiveData asset is used.
7. Specify the **Start row** and the **End row** to define which rows in the defined sheet will be used during the repetition.

   For additional information, see *Determining Data Use for ActiveData*.
8. Select the corresponding option from the **Random count** radio list to define whether the repetition should go through all specified data rows sequentially or in random order, or through a specified number of repetitions with random data rows.

   For additional information, see *Determining Data Use for ActiveData*.
9. Click **Next**.

   Tip: Use the **Build the Repeat** page to define the block of steps in the visual test that will repeat. Using the **Build the Repeat** page to define the repeating block of steps in a visual test is similar to using the **Define ActiveData Area** wizard pages to define a repeating block of code for a test script.

   The **Build the Repeat** page opens.
10. Once the steps to repeat have been defined for the ActiveData repetition logic, click **Next**. The **Summary** page opens.
11. Click **Finish**. Silk Test Workbench inserts the repetition logic steps before the first and after the last steps selected in the **Build the Repeat** page of the **Test Logic Designer** wizard. The step text for the inserted step that starts the repetition logic is as follows: `Repeat using ActiveData '[ActiveData asset name]'`

    The step text for the inserted step that ends the repetition logic is as follows: `End Repeat`

Mapping Data in an ActiveData File to Data in a Visual Test

Before a visual test can use data in an ActiveData file, data in the applicable test steps must be mapped to use data in columns of an ActiveData file. To do this, you first associate the ActiveData asset that contains

the desired ActiveData file with the visual test. Then, each step that is to use data in an ActiveData file must be mapped to a column in the Active Data file that contains the appropriate data. Use this procedure to map data in test steps to data in an ActiveData file.

Note: When data is mapped to a data column in ActiveData file, only the first record in the column is used during playback unless repetition logic using data from an ActiveData file is created for the mapped data.

1. In the visual test, select the first step containing data to be substituted with data in an ActiveData file associated with the visual test.

   Properties for the step display in the **Properties** pane. Properties can be sorted by category or alphabetical order. If repetition logic using data from an ActiveData file has already been created for the visual test, steps containing data to be mapped should be nested between `Repeat using ActiveData` and `End Repeat` steps.
2. In the **Properties** pane for the step, click in the value area of the property that contains the data to be substituted.
3. Click the **Select** button in the value area and click **ActiveData**. The **Select an ActiveData Column** dialog box opens. The **ActiveData asset** text box shows the ActiveData assets associated with the visual test. The name of the first asset in alphabetical order displays in the box.
4. In the **ActiveData asset** text box, select the name of the ActiveData asset containing the data to be mapped.

   The column names from the ActiveData file for the selected ActiveData asset display in the **Columns** list.

   - If the ActiveData file for the selected ActiveData asset is currently open in its native application (a text editor or Microsoft Excel) column names do not display in the **Columns** list. If this happens, close the native application and re-select the name of the ActiveData asset containing the data to be mapped.
   - If the **Use the first row as header** check box is checked for the ActiveData asset, the first row of data in the data file is assumed to be column headers. The data for each column in the first row then becomes the column names that display in the **Columns** list.
   - If the **Use the first row as header** check box is not checked for the ActiveData asset, the first row of data in the data file is assumed to be data. In this case, Silk Test Workbench uses sequential alphabetic characters for the names in the **Columns** list ("A", "B", "C").

   Tip: When creating ActiveData files, use named headers when possible. Seeing descriptive column names in the **Columns** list of the **Select an ActiveData Column** dialog box makes it easier to map data to the correct column in the ActiveData file.
5. In the **Columns** list, select the column name that contains the data to be substituted for the actual data used in the selected property.
6. Click **OK** to close the **Select an ActiveData Column** dialog box and map the data. Silk Test Workbench replaces the actual data with an expression that maps to the selected data in the ActiveData file. The expression syntax is as follows: `[[ActiveData asset name.data type]("Column name")` where `[ActiveData asset name.datatype]` is the name of the selected ActiveData asset and the data type of the data in the selected column. For visual tests, Silk Test Workbench automatically interprets data used in ActiveData files. If the data contains only numeric characters, the data type is Long . If the data contains any alphabetic characters, the data type is Text.

   ("*Column name*") is the name of the column to which the property's data is mapped.

   Tip: Values that are set by selecting a literal, variable, expression, or ActiveData can be reset if the visual test was saved when the original value was in place. Reset a value for a property by clicking the **Select** button in the value area and assign the original literal, variable, expression result, or ActiveData data type. The original value for the type appears as the property value.
7. Repeat the preceding steps for all other test steps containing data to be substituted with data in an ActiveData file associated with the visual test.

When data is mapped to a data column in ActiveData file, only the first record in the column is used during playback unless repetitive logic using data from an ActiveData file is created for the mapped data.

Note: Although you can use multiple ActiveData asset in the same visual test, you can only use one ActiveData asset in a single repetition statement.

ActiveData Updates in Visual Test Step Text

Silk Test Workbench updates a step's text when data in the step is mapped to data in an ActiveData file. In the step, the literal data in the **Steps** column of the **Test Steps** pane is replaced text that indicates the ActiveData asset, the type of data being substituted, and the column in the ActiveData file that contains the data to be substituted.

The following examples show how Silk Test replaces literal data in the step text with mapped data in an ActiveData file.

Example 1: Text input

The following example shows an automation step that enters the literal value `Smith` into a TextField control in a test application. The literal value is in the **text** property for the step:

```
Enter 'Smith'
```

When the literal data is substituted with data from a column named **Last Name** in an ActiveData asset named **contacts**, the step text changes as follows:

```
Enter '[[contacts].Text("Last Name")]'
```

In this case, Silk Test Workbench interprets the substituted value as text because the substituted data contains alphabetic characters, so the data type specified in the step is `Text`.

Example 2: Numeric input

The following example shows a step that moves a window to a position on the screen that is based on coordinate values in the `Left` and `Top` properties for the step:

```
Move to 235, 427
```

When the values for the `Left` and `Top` properties are substituted with data from columns named **Left Coordinate** and **Top Coordinate** in a file associated with an ActiveData asset named **coordinates**, the step text changes as follows:

```
Move to [[coordinates].Long("Left Coordinate")], [[coordinates].Long("Top
Coordinate")]
```

In this case, Silk Test Workbench interprets the value in the ActiveData file as numeric because the substituted data contains only numeric characters, so the data type specified in the step is `Long`.

Updating ActiveData File Data in a Visual Test

You can update data in any ActiveData file associated with a visual test in the visual test. Use expressions to update the ActiveData in a visual test. The **Expression Designer** lets you use data from an ActiveData file column as input, and create an expression that changes the value of the data. The updated data can then be saved back to the same column in the same ActiveData file.

When used in repetition logic for ActiveData testing, data for each row in a column of an ActiveData file can be updated through an expression, then stored to the original row in the column. To do this, you use the column in the expression, update the data in the expression (which updates data in each row of the column) then put the contents of the expression back into the same column of the ActiveData file. You can also store the expression contents to another column in any other associated visual test, or store the updated data in a local variable, where it can be used in test logic, results comments, passed to a global variable, or passed to other visual tests or scripts.

ActiveData that is updated in a visual test using expressions is automatically saved when the visual test plays back to completion. However, you can modify the visual test to save updated data immediately or at any point during playback, or prevent the automatic save.

Updating ActiveData in a Visual Test Using an Expression

You can update data in any ActiveData file associated with a visual test in the visual test. To do this, use the **Expression Designer** to specify the data and update it. Save updated data to a variable and then use the variable in the visual test, or save it back to the ActiveData file.

1. In the visual test, create any local variables that you want to use in the expression that you are creating.
2. In the visual test, associate ActiveData assets whose data columns you want to use in the expression that you are creating.
3. Select the step that precedes where you want to create the Expression step.

   Tip: You can place expressions that use data from an ActiveData file anywhere in a visual test, including outside of any repetition logic that uses ActiveData. However, if saving data updated in an expression result back to each row in an ActiveData column, this step should be placed immediately after the step that updates the data inside the repetition logic. If the Expression step is placed after the repetition logic, the expression only updates data in the last accessed row of the ActiveData file.
4. Choose **Insert** > **Expression**. The **Expression Designer** dialog box opens.
5. Use the **Expression Designer** dialog box to create an expression that updates data from a column in an ActiveData file.

   To update ActiveData in a column and put the contents back to the same column, you update the value using the **Expression Editor** in the **Expression Designer** dialog box, then put the contents of the expression back to the same column.

   For example, if you use a column named "Gender" in the data file as input in the **Expression Editor**, assuming values in the column are `string` data type and are either "M" or "Fem", the **Expression Editor** adds "ale" to the data to create "Male" or "Female". That data is then saved back to the same column as shown in the **Output** section of the dialog box.
6. After creating and testing the expression, click **OK** in the **Expression Designer** dialog box. Silk Test Workbench inserts an `Expression:` step after the selected step.

   Tip: To edit the expression after creating it, select the `Expression:` step, then select the `Expression` property for the step and click `Expression Designer` to edit the expression using the **Expression Designer** dialog box.

ActiveData that is updated in an expression is automatically saved back to the ActiveData file upon successful playback completion. However, you can create a step to save updates immediately back to the ActiveData file, or cancel the save so updated data does not get saved.

Immediately Saving Updated ActiveData in a Visual Test

Any ActiveData that is updated in an expression during visual test playback is automatically saved back to the ActiveData file when playback successfully completes. If data from various cells in an ActiveData file has been changed, Silk Test Workbench saves the updated data back to the same cells unless other cells are specified.

Rather than waiting for successful playback to save data automatically back to the ActiveData file, you can create a step to immediately save specific updated data. The step only saves selected ActiveData that has been updated to that point in the visual test. ActiveData that is updated after the step executes is saved upon successful playback, unless the automatic save is cancelled.

1. In the visual test, create any local variables to be used in the expression to be created.
2. In the visual test, associate ActiveData assets whose data columns you want available for use in the expression to be created.

3. Create expressions that update data in an ActiveData file and store the updates back to the original cells.
4. Select the last `Expression:` step in the visual test that updates data that you want to save immediately.
5. Choose **Insert** > **ActiveData** > **Save Now**. Silk Test Workbench inserts a `Save values immediately to ActiveData file` step below the selected step.
6. Select the `Save values immediately to ActiveData file` step that you just created. Properties for the step appear in the **Properties** pane.
7. Set the ActiveData category properties to specify the ActiveData asset containing the data file whose updated values are to be saved immediately.

Tip: Use **ActiveData** properties for the `Save values immediately to ActiveData file` step to change the save option or ActiveData to be saved if necessary.

Preventing Automatic Saves for ActiveData Updates in a Visual Test

Any ActiveData that is updated in an expression during visual test playback is automatically saved back to the ActiveData file when playback successfully completes. However, you can create a step to cancel the automatic saving of updated ActiveData.

Note: The step only prevents updated ActiveData from being saved up to the point this step executes. If any ActiveData is updated after this step executes, it is saved back to the ActiveData file unless the automatic save is cancelled after the updates are made.

1. In the visual test, create expressions that update data in an ActiveData file and store the updates back to the original cells.
2. Select the last `Expression:` step in the visual test that updates data that you want to prevent from being automatically saved.
3. Choose **Insert** > **ActiveData** > **Cancel Save**. Silk Test Workbench inserts a `Do not save values to ActiveData file` step below the selected step.
4. Select the `Do not save values to ActiveData file` step that you just created. Properties for the step appear in the **Properties** pane.
5. Set the **ActiveData** category properties for the step to specify the ActiveData asset containing the data file whose updated values are not to be immediately saved.

Specifying the Sheet that is Used for ActiveData in a Visual Test

When you have defined an Excel spreadsheet as the source for an ActiveData asset, and this ActiveData asset is used for ActiveData testing in a visual test, you can specify the sheet that you want to use during execution directly in the visual test:

1. Open the visual test.
2. Select the **Repeat using activedata ...** step.
3. Open the **Properties** pane for the step.
4. In the **Condition** area, type the name of the sheet into the **Sheet name** field.

   During the execution of the visual test, the sheet specified here overrides the sheet that is specified in the ActiveData asset. If you leave this field empty, the sheet that is specified in the asset is used.
5. Save the visual test.

During execution of the visual test, Silk Test Workbench throws an error if the specified sheet does not exist within the Excel spreadsheet. In this case, review the Excel spreadsheet and specify an existing sheet.

Getting the Row Count or the Column Count of an Active Data Asset

To get the row count and the column count of an active data asset, you can insert the respective `Get Row Count` and `Get Column Count` steps into a visual test that is associated with the asset.

1. You can insert these steps into a visual test in the following ways:
   - Select **Insert** > **ActiveData** > **Get Row Count** or **Insert** > **ActiveData** > **Get Column Count** in the Silk Test Workbench menu.
   - Select **Actions** > **Insert** > **ActiveData** > **Get Row Count** or **Actions** > **Insert** > **ActiveData** > **Get Column Count** in the **Test Steps** pane of the visual test.
   - Right-click into the **Test Steps** pane of the visual test and select **Insert** > **ActiveData** > **Get Row Count** or **Insert** > **ActiveData** > **Get Column Count**.
2. In the **Properties** window for these steps, perform the following actions:
   a) Select the associated active data asset to use.
   b) Select whether to run a row count or a column count.
   c) Select what variable to put the result in.

      The result will always be saved in the *st_LastReturnValue* reserved variable. It is not required to select a local variable. The result is also saved in the **Result Detail** column of the test step in the **Details** pane of the test result.

If the `Get Row Count` or `Get Column Count` steps are used within a `Repeat using ActiveData` loop, the number of rows is based on the start and end row that is configured in the `Repeat using ActiveData` step. For example, if the active data file has 10 rows, but the `Repeat using ActiveData` loop is configured to use only the rows 2 to 6, the `Get Row Count` step returns 5.

If the steps are used outside of a `Repeat using ActiveData` loop, they return the actual number of rows and column in the active data file.

Using ActiveData in Scripts

The process of creating a script that uses ActiveData is independent of the record feature. Test data is stored in and retrieved from outside the Silk Test Workbench database, using external files. Test data can reside in Excel Spreadsheet files (.xls and .xlsx), text files (.txt), or comma separated value (.csv) files. Storing test data in external files lets you reuse test data in more than one ActiveData asset and in any visual test or script.

Note: If Microsoft Excel® is not installed, all .xls files are in read-only mode. To use a .xlsx file with Silk Test Workbench, install Microsoft Excel® 2007 or later. If Microsoft Excel® 2007 or later is not installed, you can install *2007 Office System Driver: Data Connectivity Components* to use .xlsx files in read-only mode.

Note: If an error occurs while opening an ActiveData file, use error handling to continue the test execution.

Creating an ActiveData Test Asset

When creating an ActiveData asset, you either create a new file to contain the data used by the asset, or select an existing file that contains data to be used. To use an existing file, make sure the file exists and is available prior to creating the asset.

1. Choose **View** > **Asset Browser**.
2. Under **Asset Types**, right-click **ActiveData** and choose **New ActiveData**. The **ActiveData** window opens.
3. Type a descriptive name for the asset in the **Name** text box.

   ActiveData asset names must follow Silk Test Workbench asset naming conventions.

Tip: Giving the ActiveData asset the same name as the ActiveData file used for the asset helps to quickly identify the asset for test use.

4. Click **Browse** to select an existing file to use as the ActiveData file, or click **New** to create a new file to use as the ActiveData file. The ActiveData file path and file name display in the **File** text box.
5. *Optional*: Type a description for the asset in the **Description** text box.

   Use the description to indicate the type of file used as the data file, the type of testing being performed by the ActiveData test for which the asset is being created, or the application being tested with the ActiveData test. The description displays in the **Asset Browser** for any ActiveData assets that include a description.

   Note: If an empty data file is required, click **Save** and **Close** to save any changes and close the **ActiveData** setup window.
6. Continue creating the ActiveData asset by setting read options for the data file being used by the ActiveData asset.
7. Continue creating the ActiveData asset by editing the data in the ActiveData file.
8. Click **Save** to save any changes, or click **Save and Close** to save any changes and close the **ActiveData** window.

The new asset displays in the ActiveData asset listing in the **Asset Browser**. You can use it in ActiveData testing.

Including ActiveData in a Script

Before you begin, create the ActiveData asset that contains the data that you want to use or that you want to write data to.

Map the data from the ActiveData asset to the script that you want to use. Additionally, you can add rows or columns to the ActiveData asset, and add data to the ActiveData asset from the script.

1. Record or manually script the actions that you want to test.

   For more information, see *Recording a Script*.

   For example, a script that records a first name, last name, and phone number in a Web application might look like the following:

```
Public Sub Main()
  With _desktop.BrowserWindow("/BrowserApplication//BrowserWindow")
    .DomTextField("@id='txtFirstName'").SetText("Pat")
    .DomTextField("@id='txtLastName'").SetText("Smith")
    .DomTextField("@id='txtPhoneNumber'").SetText("555-121-3434")
    .DomButton("@id='btnAdd'").Select()
  End With
```

2. Use the `LoadActiveData` command to specify the ActiveData asset name.

   Optionally, you can specify a start and end row in the ActiveData file, random rows to use, and whether data is read only.

```
Workbench.LoadActiveData("activeDataName", [start row,
   end row, randomcount, readonly])
```

3. To replace the text that you recorded with text from the ActiveData asset, create a declaration for each corresponding row name in the ActiveData file and then map the declaration to the appropriate code in the script.
   For example, to modify the script from the first step to use all the data available in the *PhoneBookData* ActiveData asset, you might change the script to look like the following:

```
Public Sub Main()
  AddAllToPhoneBook()            'Reads all records in order
End Sub

Public Sub AddAllToPhoneBook()
  Dim data As ActiveData = Workbench.LoadActiveData( "PhoneBookData" )
  Dim row As ActiveDataRow
```

```
  With _desktop.BrowserWindow("/BrowserApplication//BrowserWindow")

    For Each row In data

    Dim FirstName As String = row.GetString("fname")
    Dim LastName As String = row.GetString("lname")
    Dim PhoneNumber As String = row.GetString("phonenumber")

    .DomTextField("@id='txtFirstName'").SetText(FirstName)
    .DomTextField("@id='txtLastName'").SetText(LastName)
    .DomTextField("@id='txtPhoneNumber'").SetText(PhoneNumber)

    .DomButton("@id='btnAdd'").Select()
    Next
  End With
End Sub
```

When you playback the script, all the declared data from the ActiveData asset is included in the script.

4. To add an additional column to the existing ActiveData asset, declare the ActiveData asset, and specify the ActiveData asset name and the column name.
   For example, to add a new column called "Occupation" to the PhoneBookData asset, type the following:

```
Public Sub Main()

  AddColumnToDataFile( "occupation" )
End Sub

Public Sub AddColumnToDataFile( name As String )
  Dim data As ActiveData = Workbench.LoadActiveData( "PhoneBookData" )
  data.AddColumn( "Occupation", name )
  data.Save()
End Sub
End Module
```

When you playback the script, the new column is added to the file specified by the ActiveData asset.

5. To add an additional row to the existing ActiveData asset, declare the ActiveData asset, and specify the ActiveData asset name and the row name.
   For example, to add a new row of data for ID number, type the following:

```
Public Sub Main()

  InsertNewNumberToDataFileWithOccupation( 108, "Jay", "Jones", "5551219",
_
    "QA Analyst" )

End Sub

Public Sub InsertNewNumberToDataFileWithOccupation( id As Integer, _
    firstName As String, lastName As String, phoneNumber As String, _
    occupation As String )
  Dim data As ActiveData = Workbench.LoadActiveData( "PhoneBookData" )
  Dim row As ActiveDataRow = data.AddRow()

  row.SetLong( "id", id )
  row.SetString( "fname", firstName )
  row.SetString( "lname", lastName )
  row.SetString( "number", phoneNumber )
  row.SetString( "occupation", occupation )

  data.Save()

End Sub
End Module
```

When you playback the script, the new row is added to the file specified by the ActiveData asset.

6. To add the value of a variable into a specific cell, you can add the following code sample to your script:

```
Dim row As ActiveDataRow = data.Item(1)
row.SetString("Columnname" , Variable)
```

Specifying the Sheet that is Used for ActiveData in a VB .NET Script

When you have defined an Excel spreadsheet as the source for an ActiveData asset, and this ActiveData asset is used for ActiveData testing in a VB. NET script, you can use the `sheetName` parameter of the `LoadActiveData` method to specify the sheet that you want to use during execution. For additional information, see *LoadActiveData Method*.

During execution of the VB .NET script, Silk Test Workbench throws an error if the specified sheet does not exist within the Excel spreadsheet. In this case, review the Excel spreadsheet and specify an existing sheet.

Enhancing Visual Tests

This section lists ways to modify visual tests to perform more advanced functions.

Changing the Action that is Performed in a Visual Test Step

Use the **Select Actions** dialog box to change the action that is performed on a control in a step of a visual test.

The list includes the following columns:

Click an action to select it, then click **OK** to use it in the test step as the action performed against the control.

Note: Most actions use additional parameters to perform the selected action in a specific manner against the control. The parameters for the action appear as properties in the **Parameters** category and can be set in the **Properties** window.

1. Open the visual test that contains the action that you want to change.
2. In the **Test Steps** pane, select the step that includes that action that you want to change.
3. Open the **Properties** pane.
4. In the **Action** section of the pane, click on **Method**.
5. Click **Select Action**. The **Select Action** dialog box opens, listing all allowable actions for the UI control with which the current test step interacts. The dialog box can be resized to see the text in each column of the list.
6. Click on the action that you want to perform on the UI control.
7. Click **OK**.

Inserting a Comment into a Visual Test

Comments allow you to provide supplemental information about the steps in a visual test. You can insert comments as steps into existing visual tests.

1. In the **Test Steps** pane of the visual test, select the step after which you want to insert the comment.
2. Choose **Insert** > **Comment**. The `Comment` step is added to the **Test Step** pane and properties for this step display in the **Properties** pane.

3. In the **Properties** pane, type the text for the comment into the **Comment** field.

Flags

Flags help you remind yourself or another team member about an issue in a visual test or result. Flagging an issue adds either a **Flag** icon or **Assigned Flag** icon depending on the type of flag you insert.

Flags contain the following information:

- The name of the user who created or modified the flag.
- The date of when the flag was created or modified.
- A description of the flag (optional).
- The name of the user to whom the flag is assigned (optional).
- The project in which the visual test or result is saved (optional).

The **Flag** icon appears in the **Flags** column of the **Test Steps** pane. When you move your pointer over the flag, a ToolTip displays the contents of the flag.

Insert flags using either of the following methods:

- Select a step in the **Test Steps** pane of either a visual test or result and insert a flag.

  Optionally, you can create a description for the flag and assign it to another user. Assigned flags appear in the selected step of the visual test or result and in the **Flags** pane of the **Start Screen** of the user to which the flag is assigned. From the **Start Screen**, you can quickly view all flags assigned to you and open the visual test or result that the flag appears in by double-clicking the flag.
- Create verification logic in a visual test that failure during playback sends a flag to the result of the visual test.

  In the result, the flag is inserted in the step containing the verification logic. This step appears in the **Test Steps** pane of the **Failed** tab, **Flags** tab, and **Details** tab of the result. Optionally, you can create a description for the flag in the wizard. By default, this flag is automatically assigned to the current user. If you want to assign it to someone else, you must reassign it after playback from the result.

Note: Flags are only supported for visual tests, not for Visual Basic .NET scripts.

Inserting a Flag in the Test Steps Pane

Insert a flag to help you keep in mind an issue in a visual test or result.

1. Open the visual test or result in which you want to insert a flag.
2. In the **Test Steps** pane of a visual test or the **Passed** tab, **Failed** tab, **Flags** tab, or **Details** tab of a result, select the step in which to insert a flag.
3. Click **Actions** > **Flag** > **Insert**.

   You can also access the **Flag** dialog box, by double-clicking the cell in the **Flag** column of the step in which you want to insert a flag.

   The **Flag** dialog box opens.
4. *Optional:* Enter a description of the flag in the **Description** text box.

   The description displays in a ToolTip when you move your pointer over the **Flag** or **Assigned Flag** icon.
5. *Optional:* Assign the flag to another user by checking the **Show on Start Screen and assign to** check box and selecting a user. The flag appears on the **Start Screen** of the assigned user.
6. Click **OK**. Depending on the options you selected in the **Flag** dialog box, a **Flag** icon or **Assigned Flag** icon appears in the **Flag** column of the selected step. For a result flag, the flag is saved immediately after you click **OK**. For a visual test flag, you must save the visual test to save the flag.

Inserting a Flag in a Result

If the verification logic fails during playback, Silk Test Workbench sends a flag to the result of the visual test.

1. Create or record a visual test.
2. Insert verification logic using the **Test Logic Designer** wizard.
3. Check the **If the verification fails, create a 'flag' in the result detail** check box and specify a flag description.
4. Playback the visual test. If the verification logic fails during playback, Silk Test Workbench sends a flag to the result of the visual test. In the result, the flag is inserted in the step containing the verification logic, which appears in the **Failed** tab, **Flags** tab, and **Details** tab of the **Result** window.

Viewing a Flag

View the contents of a flag.

1. Open the visual test or result in which you want to view a flag.

   In the **Test Steps** pane of a visual test or the **Passed** tab, **Failed** tab, **Flags** tab, or **Details** tab of a result, a flagged step is indicated by a **Flag** icon or **Assigned Flag** icon in the **Flag** column.
2. Move your pointer over the **Flag** or **Assigned Flag** icon. A ToolTip shows the contents of the flag.

Viewing a Flag From the Start Screen

View the contents of a flag from the **Start Screen**.

1. Open the **Start Screen**. By default, flags appear in the **Flags** pane grouped by the date that they were created and the asset in which they appear.

   Note: You can change the display of the **Flags** pane by modifying the **Start Screen Flag** options, which you can access from the **Start Screen** by clicking **Customize** located below the **Help** pane of the **Start Screen**.
2. Navigate to the flag that you want to see and move your pointer over the **Assigned Flag** icon. A ToolTip displays the flag contents.
3. To open the visual test or result in which the flag is inserted, double-click the **Assigned Flag** icon.

Editing a Flag

Modify the flag contents to meet your needs.

1. Open the visual test or result that contains the flag that you want to edit.
2. In the **Test Steps** pane of a visual test or the **Passed** tab, **Failed** tab, **Flags** tab, or **Details** tab of a result, select the step that contains the flag to edit.
3. Click **Actions** > **Flag** > **Edit**. The **Flag** pane opens. You can also double-click the **Flag** or **Assigned Flag** icon to open the **Flag** dialog box.
4. *Optional:* Enter a description of the flag in the **Description** text box.

   The description displays in a ToolTip when you move your pointer over the **Flag** or **Assigned Flag** icon.
5. *Optional:* Assign the flag to another user by checking the **Show on Start Screen and assign to** check box and selecting a user. The flag appears on the **Start Screen** of the assigned user.
6. Click **OK**. For a result flag, the flag is saved immediately after you click **OK**. For a visual test flag, you must save the visual test to save the flag.

Removing a Flag

Remove a flag when it is no longer necessary.

1. Open the visual test or result that contains the flag that you want to remove.
2. In the **Test Steps** pane of a visual test or the **Passed** tab, **Failed** tab, **Flags** tab, or **Details** tab of a result, select the step that contains the flag you want to remove.
3. Click **Actions** > **Flag** > **Remove**. A message appears asking you if you want to remove the flag.
4. Click **Yes**. For a result flag, the flag is removed immediately after you click **OK**. For a visual test flag, you must save the visual test to permanently remove the flag.

Adding Test Logic

You can enhance the power and flexibility of a visual test by including test logic that expands the basic capability of a recorded visual test. By inserting test logic, you can dynamically alter the execution flow of the visual test. For example, you can insert test logic into your visual test that evaluates a user-defined condition, and then performs one of the following actions based on the result of the condition:

- Runs or repeats a sequence of test steps in a visual test
- Runs a visual test from a visual test
- Sends a pass/fail message and flag regarding the playback status of a test step to the result of a visual test.

To accomplish these actions, Silk Test Workbench provides several different types of test logic, which are categorized as follows:

- *Decision Logic*: Logic that evaluates a user-defined condition and then runs a sequence of test steps depending on the return value of the condition.
- *Repetition Logic*: Logic that repeats a sequence of test steps a set number or times or while a user-defined condition is true.
- *Verification Logic*: Logic that evaluates a user-defined condition and then sends a pass/fail message and flag to the result of a visual test.
- *Error Handling Logic*: Logic that performs a user-defined action in the event of an error during playback.

Silk Test Workbench provides the following tools to insert test logic into your visual test.

- The *Test Logic Designer Wizard*.
- The *Logic Toolbox*.
- The decision test steps *If*, *Else If*, and *Else*.

Decision Logic

Decision logic evaluates a user-defined condition, and then runs a sequence of test steps depending on the return value of the condition. The condition used in decision logic compares two values and returns a value based on the comparison. For more information, see *Condition Designer*. The values in the condition can be the value of a property of a control in the test application, the value of a variable used in the visual test, or a literal value. For more information, see *Test Logic Designer: Select a Logic Type (Decision)*.

Decision logic consists of the following elements:

```
If <condition=TRUE> Then
  __<Run sequence of test steps>
End If
```

For example, the condition in the following decision logic compares the variable *propertyVar* with the literal value "4". If the variable propertyVar is greater than the literal value "4", then steps 5 through 9 are run:

```
If propertyVar > 4 Then
  __Do Step 5 To Step 9
End If
```

Additionally, you can create more complex decision logic that can be nested to as many levels as you need by using `Else IF` and `Else`.

```
If <condition=TRUE> Then
  __<Run sequence of test steps>
```

```
Else If <condition=TRUE> Then
  __<Run sequence of test steps>
Else
  __<Run sequence of test steps>
End If
```

For example, the condition in the following decision logic compares the variable *propertyVar* with the literal value "4". If the variable *propertyVar* is greater than the literal value "4", then steps 5 through 9 are run. If it is less than "4", then steps 10 through 14 are run. If it is equal to "4", then step 15 is run:

```
If propertyVar > 4 Then
  __Run Step 5 to Step 9
Else If propertyVar < 4 Then
  __Run Step 10 to Step 14
Else
  __Run Step 15
End If
```

You can create decision logic using either the **Test Logic Designer** wizard, the Logic Toolbox , or by using `If...Else If...Else` test steps.

Adding Decision Logic to a Visual Test

This topic describes how you can use the **Test Logic Designer** to add decision logic to a visual test. For information on how to add decision logic test steps directly to the visual test, see *Inserting Decision Logic Using Decision Test Steps*.

1. Open the visual test.
2. Right-click on the test step or the sequence of test steps that you want to include in your test logic.
3. Choose **Insert** > **Test Logic** > **Decision**. The **Test Logic Designer** wizard opens.
4. Click **Next**.
5. From the **Select a Logic Type** page, select the type of decision logic.
   - **The property of a control** – Creates decision logic that runs a sequence of steps based on the property of a control in your test application. For example:
     ```
     If propertyValue > 2 then Do step 3 To step 4 End If
     ```
   - **Whether a control exists** - Creates decision logic that runs a sequence of steps based on whether a specified control exists or the specified control does not exist. For example:
     ```
     If "Button" Exists Is Equal to True Do Step 3 To Step 4 End If
     ```
   - **A variable defined in this visual test** – Creates decision logic that runs a sequence of steps based on the contents of a local variable defined in your visual test. For example:
     ```
     If stringVar = "red" then Do step 3 to step 4 End If
     ```
   - **Advanced logic** – Skips the *Defining the Condition* step of the wizard and creates advanced decision logic using the **Condition Designer**. For example:
     ```
     If stringVar = "red" AND numberVar = 10 then Do step 3 To step 4 End If
     ```
6. Click **Next**.
7. If you have selected **The property of a control**, the **Define a property-based condition** page opens.
   a) Expand the **Identify a Control** menu to the right of the **Name of control** field.
   b) Expand the **Identify a control** menu and select one of the following:
      - **Application Under Test** – Click this button to identify a visible control directly from the application under test.
      - **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
      - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.

c) In the **Select a property** table, select the property.
d) From the **Select the condition** list, select the conditional logic to apply to the property of the control.
e) In the **Expected value** field, specify the value that the property is expected to have.
f) Click **Next**.

8. If you have selected **Whether a control exists**, the **Define an exists condition for the control** page opens.
   a) Expand the **Identify a control** menu and select one of the following:
      - **Application Under Test** – Click this button to identify a visible control directly from the application under test.
      - **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
      - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.
   b) Select **Control exists** to verify that the control exists.
   c) Select **Control does not exist** to verify that the control does not exist.
   d) Type a value into the **Timeout** field to specify how long Silk Test Workbench should wait for the control to appear or to disappear.

      The default timeout is 0, which means Silk Test Workbench does not wait for the control to appear or to disappear.
   e) Click **Next**.
9. If you have selected **A variable defined in this visual test**, the **Define a variable-based condition** page opens.
   a) Select a pre-defined local variable from the **Select the variable** list.

      For a variable to appear in this list, you must include the local variable in your visual test. To add a new local variable to the visual test, click **New**.

      The **Evaluate as** field, displays the data type of the selected variable.
   b) Select the comparison operator to apply to the variable from the **Select the condition** list.
   c) In the **Expected value** field, specify the value that the variable is expected to have.
   d) Click **Next**.
10. If you have selected **Advanced logic**, click **Next**.
11. The **Build the Decision** page opens. Specify the conditional logic and sequence of steps to run when the result of the conditional logic is true.

    Tip: Rest your pointer on a link to display a ToolTip containing the full link description.

    a) Click the link to the right of the **If** statement to specify the conditional logic in the *Condition Designer*.
    b) Click the link to the right of the **Do step** statement to specify the first step in the sequence of steps to run.
    c) Click the link to the right of the **To step** statement to specify the last step in the sequence of steps to run.
    d) Click **Add Else If** to insert additional conditional logic.

       Click the provided links to open the **Condition Designer** and specify the sequence of steps to run when the condition is true.
    e) Click **Add Else** to insert additional conditional logic.

       Use the provided links to specify the sequence of steps to run when the condition is false.
    f) Click **Delete** to remove any additional conditional logic and sequence of steps.

       This button only appears after you insert additional conditional logic.
12. Click **Next**. The **Summary** page displays the test logic that you have defined. Uncheck the check box at the bottom of the page to prevent the **Summary** page from appearing on subsequent uses of the **Test**

Logic Designer wizard. Checking or un-checking this check box updates the **Show Test Logic Designer summary screen** general option. For more information, see *Modifying General Options*.

13. Click **Finish** to insert the test logic into the test steps of your visual test. To modify the test logic, click **Back** and make the desired changes. The test logic is added to the visual test.

Adding Decision Logic Using Decision Test Steps

You can quickly create decision logic by inserting `If`, `Else If`, and `Else` test steps, which provide an alternate method from the **Test Logic Designer** wizard for creating decision logic. Additionally, you can quickly modify existing decision logic without using the **Test Logic Designer** wizard.

`Else If`, and `Else` test steps allow you to create complex decision logic that can be nested to as many levels as you need.

```
If <condition=TRUE> Then
  __<Run sequence of test steps>
Else If <condition=TRUE> Then
  __<Run sequence of test steps>
Else
  __<Run sequence of test steps>
End If
```

For example, the condition in the following decision logic compares the variable *propertyVar* with the literal value "4". If the variable *propertyVar* is greater than the literal value "4", then steps 5 through 9 are run. If it is less than "4", then steps 10 through 14 are run. If it is equal to "4", then step 15 is run:

```
If propertyVar > 4 Then
  __Run Step 5 to Step 9
Else If propertyVar < 4 Then
  __Run Step 10 to Step 14
Else
Run Step 15
End If
```

To quickly insert decision test steps into a visual test:

1. Open the visual test and select the test steps that you want to include in the decision logic.

   To select a sequence of test steps, select the first step, press and hold the `Shift` key, and then select the last test step in the sequence. Any test steps between the first and last test step are automatically selected.
2. Click **Insert an If Type Logic Item** on the toolbar.

   The selected test steps are bracketed by an `If` test step and an `End If` test step.
3. *Optional*: To insert an `Else If` or `Else` test step in the If...End If logic, select the test step that you want the `Else If` or `Else` test step to appear after, and click either **Insert an Else If Type Logic Item** or **Insert an Else Type Logic Item**.

Editing Decision Logic Test Steps

After creating test logic steps using the **Test Logic Designer** wizard or the **Condition Designer**, you can edit the test steps directly from the **Properties** pane.

Note: Only the `If` and `Else If` test steps contain the **Condition** property. The `Else` and `End If` test steps do not.

1. Open the visual test that contains the test step that you want to edit.
2. In the **Test Steps** window of the visual test, select the test step that you want to edit. The properties and associated property values for the selected test step appear listed by property category in the **Properties** pane.
3. Edit the available condition property or miscellaneous properties.

Repetition Logic

Repetition logic repeats a sequence of test steps a set number of times while or until a user-defined condition is true. The condition used in repetition logic compares two values and returns a value based on the comparison. For more information, see *Condition Designer*. The values in the condition can be either the value of a property of a control in the test application, the value of a variable used in the visual test, a literal value, or a value from an ActiveData asset.

Repetition logic that repeats a sequence of test steps a set number of times consists of the following elements:

```
Repeat <number of times>
  __<Run sequence of test steps>
End Repeat
```

For example, the sequence of test steps in the following repetition logic is repeated from step 10 to step 15:

```
Repeat 10 times
  __Do step 10
  __To step 15
End If
```

Repetition logic that repeats a sequence of test steps while, or until a user-defined condition is true consists of the following elements:

```
Repeat while <condition=TRUE>
  __<Run sequence of test steps>
End Repeat
```

Or, you can specify the condition at the end of the loop:

```
Repeat
  __<Run sequence of test steps>
End Repeat while <condition=TRUE>
```

For example, the sequence of test steps in the following repetition logic repeats while the value of the variable *propertyVar* is less than the literal value "4". If *propertyVar* is greater than or equal to "4", the loop is never run.

```
Repeat while propertyVar < 4
  __Do Step 10
  __To Step 14
End Repeat
```

Or, you can specify the condition at the end of the loop. If *propertyVar* is greater than or equal to "4", the loop is run once:

```
Repeat
  __Do Step 10
  __To Step 14
End Repeat while propertyVar < 4
```

You can create repetition logic using either the **Test Logic Designer** wizard or the Logic Toolbox.

Adding Repetition Logic to a Visual Test

This topic describes how you can use the **Test Logic Designer** to add repetition logic to a visual test.

1. Open the visual test.
2. Right-click on the test step or the sequence of test steps that you want to repeat.
3. Choose **Insert** > **Test Logic** > **Repetition**. The **Test Logic Designer** wizard opens.
4. Click **Next**.
5. From the **Select a Logic Type** page, select the type of repetition logic.

   Select a type of repetition logic based on:

- **The property of a control** – Creates repetition logic that repeats a sequence of steps based on the property of a control in your test application. For example:

  ```
  Repeat While propertyValue = 3 Do step 3 To step 4 End Repeat
  ```
- **Whether a control exists** - Creates repetition logic that repeats a sequence of steps based on whether a specified control exists or the specified control does not exist. For example:

  ```
  Repeat While "Button" Exists Is Equal to True Do Step 3 To Step 4 End
  Repeat
  ```
- **A variable defined in this visual test** – Creates repetition logic that repeats a sequence of steps based on the contents of a local variable defined in your visual test. For example:

  ```
  Repeat While numberVar < 5 Do step 3 To step 5 End Repeat
  ```
- **Repeating a sequence of steps a set number of times** - Creates repetition logic that repeats a sequence of steps a set number of times. For example:

  ```
  Repeat this loop 10 times Do step 3 To step 7 End Repeat
  ```
- **The contents of a list or combo box** - Creates repetition logic that repeats a sequence of steps for each item in a list or combo box. For example:

  ```
  Repeat for each item in ListA Do step 3 To step 5 End Repeat
  ```
- **Repeating a sequence of steps using data from an ActiveData file** - Creates repetition logic to run a sequence of steps based on the data in an ActiveData file used in your visual test. For example:

  ```
  Repeat using ActiveData file testData Do step 3 To step 5 End Repeat
  ```
- **Advanced logic** – Skips the *Defining the Condition* step of the wizard and creates advanced repetition logic using the **Condition Designer**. For example:

  ```
   Repeat While propertyValue = red OR stringVar = red Do step 3 To step 5
  End Repeat
  ```

6. Click **Next**.
7. If you have selected **The property of a control**, the **Define a property-based condition** page opens.
   a) Expand the **Identify a Control** menu to the right of the **Name of control** field.
   b) Expand the **Identify a control** menu and select one of the following:
      - **Application Under Test** – Click this button to identify a visible control directly from the application under test.
      - **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
      - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.
   c) In the **Select a property** table, select the property.
   d) From the **Select the condition** list, select the conditional logic to apply to the property of the control.
   e) In the **Expected value** field, specify the value that the property is expected to have.
   f) Click **Next**.
8. If you have selected **Whether a control exists**, the **Define an exists condition for the control** page opens.
   a) Expand the **Identify a control** menu and select one of the following:
      - **Application Under Test** – Click this button to identify a visible control directly from the application under test.
      - **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
      - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.
   b) Select **Control exists** to verify that the control exists.

c) Select **Control does not exist** to verify that the control does not exist.
d) Type a value into the **Timeout** field to specify how long Silk Test Workbench should wait for the control to appear or to disappear.

   The default timeout is 0, which means Silk Test Workbench does not wait for the control to appear or to disappear.
e) Click **Next**.

9. If you have selected **A variable defined in this visual test**, the **Define a variable-based condition** page opens.
a) Select a pre-defined local variable from the **Select the variable** list.

   For a variable to appear in this list, you must include the local variable in your visual test. To add a new local variable to the visual test, click **New**.

   The **Evaluate as** field, displays the data type of the selected variable.
b) Select the comparison operator to apply to the variable from the **Select the condition** list.
c) In the **Expected value** field, specify the value that the variable is expected to have.
d) Click **Next**.

10. If you have selected **Repeating a sequence of steps a set number of times**, the **Define the Number of Times to Repeat** page opens.
a) Specify the number of times to repeat a sequence of steps by entering a numerical value between 1 and 32767.
b) Click **Next**.

11. If you have selected **The contents of a list or combo box**, the **Define the repeat based on a control** page opens.
a) Expand the **Identify a control** menu and select one of the following:

   - **Application Under Test** – Click this button to identify a visible control directly from the application under test.
   - **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
   - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.

   For either type of control, you can create repetition logic that uses the value of each item in a list or combo box. For example, you can create a repeat loop that iterates through the values in a list or combo box and inserts the value of each item into a text box in your test application. In this way, you can quickly test the entire contents of a list or combo box.
b) Click **Next**.

12. If you have selected **Repeating a sequence of steps using data from an ActiveData file**, the **Define the ActiveData asset to use** page opens.

Use the **Define the ActiveData Asset to use** page to define the ActiveData asset and the retrieval method of the data in the ActiveData asset to use in your repetition logic.

a) Select an **ActiveData asset** from the list.

   You must first include an ActiveData asset in your visual test to make it available in this list.
b) Select the sheet in the data file that you want to use from the **Sheet name** list.

   By default, the sheet that is specified in the ActiveData asset is used.
c) Type the number of the first row of data in your ActiveData file to use into the **Start row** field.
d) Check the **Start at last row containing data** check box to start at the last row of data in your ActiveData file.

   Silk Test Workbench automatically detects the last row in your file and reverses the iteration through the rows in your ActiveData file. For example, if your ActiveData file contains 10 rows and you select this option, Silk Test Workbench iterates starting with row 10, then row 9, row 8, and so on.

e) Type the number of the last row of data in your ActiveData file into the **End row** field.
f) Check the **End at last row containing data** check box to end at the last row of data in your ActiveData file.

Silk Test Workbench automatically detects the last row in your file.

Tip: If the **Start at last row containing data** or **End at last row containing data** check boxes are checked, the `Start Row` or `End Row` property values for the `Repeat` step are set to -1.

g) Select in which order the rows should be processed:

- **Get all rows in sequence** – Select this option to iterate through the selected rows in your ActiveData file in sequence from the first specified row to the last specified row.
- **Get all rows in random order** – Select this option to randomly iterate through all of the rows in your ActiveData file.
- **Get X random rows** – Select this option and type the number of rows in your ActiveData file to randomly iterate through.

h) Click **Next**.

13. If you have selected **Advanced logic**, click **Next**.

14. The **Build the Repeat** page opens. Specify the sequence of steps to repeat.

Tip: Rest your pointer on a link to display a ToolTip containing the full link description.

a) For a "repeat while", select whether you want the condition for the repeat to be checked at the start or at the end of the loop.
b) For a "repeat while", click the link to the right of the **Repeat while** statement to specify the conditional logic in the *Condition Designer*.
c) Click the link to the right of the **Do step** statement to specify the first step in the sequence of steps to repeat.
d) Click the link to the right of the **To step** statement to specify the last step in the sequence of steps to repeat.

15. Click **Next**. The **Summary** page displays the test logic that you have defined. Uncheck the check box at the bottom of the page to prevent the **Summary** page from appearing on subsequent uses of the **Test Logic Designer** wizard. Checking or un-checking this check box updates the **Show Test Logic Designer summary screen** general option. For more information, see *Modifying General Options*.

16. Click **Finish** to insert the test logic into the test steps of your visual test. To modify the test logic, click **Back** and make the desired changes. The test logic is added to the visual test.

Creating Repetition Logic Based on the Contents of a List or Combo Box

Identify a list or combo box control to use in your repetition logic. For either type of control, you can create repetition logic that uses the value of each item in a list or combo box. For example, you can create a repeat loop that iterates through the values in a list or combo box and inserts the value of each item into a text box in your test application. In this way, you can quickly test the entire contents of a list or combo box.

1. Record a visual test.
2. Use the **Test Logic** Wizard to create repetition logic based on a list or combo box control.
3. Modify the test step that you want to repeat using the contents of a list or combo box by replacing the literal value with the reserved variable *st_RepeatListValue*.

   The reserved variable, *st_RepeatListValue*, is used to store the value of each item in the list or combo box for each iteration of the repeat loop. *st_RepeatListValue* returns a compound value that contains both the index and text of the item.
4. To modify the test step, select the test step that you want to repeat.
5. In the **Properties** window, select the automation property of the test step you want to modify.
6. Click **Select** and then select **Variable**. The **Select a Variable** dialog box opens.

7. Select **Reserved** from the **Variable type** list, and then select **st_RepeatListValue**.
8. Click **OK**. Silk Test Workbench updates the test step text and property value with the selected variable name.
9. Run the visual test.

Editing Repetition Logic Test Steps

After creating test logic steps using the **Test Logic Designer** wizard or the **Condition Designer**, you can edit the test steps directly from the **Properties** pane.

1. Open the visual test that contains the test step that you want to edit.
2. In the **Test Steps** window of the visual test, select the repetition logic test step that you want to edit. The properties and associated property values for the selected test step appear listed by property category in the **Properties** pane.
3. Edit the available condition properties or miscellaneous properties.

   The `Repeat` test step contains both condition properties and miscellaneous properties. The `End Repeat` contains only miscellaneous properties.

Verification Logic

Verification logic evaluates a user-defined condition, and then sends a pass/fail message to the result of a visual test or script. Optionally, you can also flag a failed test step to appear in the result of a visual test. The condition used in verification logic compares two values and returns a value based on the comparison. For more information, see *Condition Designer*. The values in the condition can be either the value of a property of a control in the test application, the value of a variable used in the visual test, or the time taken to run a sequence of test steps.

Verification logic consists of the following elements:

```
If <condition=TRUE> Then
  __<Send pass message to results>
Else
  __<Send fail message/flag to results>
End If
```

For example, the condition in the following decision logic compares the variable *propertyVar* with the literal value "4". If the variable *propertyVar* is greater than the literal value "4", then a pass message is sent to the results. If the value is less than "4", a fail message and flag is sent:

```
If propertyVar > 4 Then
  __<Send pass message>
Else
  __<Send fail message & flag>
End If
```

You can create verification logic using either the **Test Logic Designer** wizard or the Logic Toolbox. You may also create verification logic during the recording of a visual test.

Adding Verification Logic to a Visual Test

This topic describes how you can use the **Test Logic Designer** to add verification logic to a visual test. Any type of verification logic sends a user-defined pass/fail message and a flag (optional) to the result of your visual test.

1. Open the visual test.
2. Right-click on the test step which includes the object that you want to verify.
3. Choose **Insert** > **Test Logic** > **Verification**. The **Test Logic Designer** wizard opens.
4. Click **Next**.

5. From the **Select a Logic Type** page, select the type of verification logic.

   Select a type of verification logic based on:

   - **The property of a control** – Creates verification logic based on the value of a property of a control. For example:
     ```
     Verify "Button"."Text" Is Equal to "Cancel"
     ```
   - **Whether a control exists** - Creates verification logic that checks if the specified control exists or if the specified control does not exist. For example:
     ```
     Verify "Button" Exists Is Equal to True
     ```
   - **A variable defined in this visual test** – Creates verification logic based on the contents of a local variable in your visual test. For example:
     ```
     Verify st_LastValidation Is Equal to False
     ```
   - **The contents of a Rumba screen** - Creates verification logic to compare the contents of a Rumba screen.
   - **A verification asset** - Creates verification logic to compare the contents of the UI against an asset. Currently, Silk Test Workbench supports only image verifications as verification assets. For additional information, see *Asset Types*. For example:
     ```
     Run the Image Verification called 'Untitled_1'
     ```
   - **The time taken to run a sequence of steps** - Creates verification logic based on the time taken to run a sequence of steps. For example:
     ```
     Verify timer number '1' is between 0 and 2 seconds Do Step 3 To Step 4
     ```
6. Click **Next**.
7. If you have selected **The property of a control**, the **Define a property-based condition** page opens.
   a) Expand the **Identify a control** menu and select one of the following:
      - **Application Under Test** – Click this button to identify a visible control directly from the application under test.
      - **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
      - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.

   b) Additionally, you can also type the full locator of the control into the **Name of control** field and click **Refresh**. If the control is found in the application under test, the **Select a property** is populated with the properties of the control.
   c) In the **Select a property** table, select the property.
   d) From the **Select the condition** list, select the conditional logic to apply to the property of the control.
   e) In the **Expected value** field, specify the value that the property is expected to have.
   f) Click **Next**.
8. If you have selected **Whether a control exists**, the **Define an exists condition for the control** page opens.
   a) Expand the **Identify a control** menu and select one of the following:
      - **Application Under Test** – Click this button to identify a visible control directly from the application under test.
      - **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
      - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.

   b) Select **Control exists** to verify that the control exists.
   c) Select **Control does not exist** to verify that the control does not exist.

d) Type a value into the **Timeout** field to specify how long Silk Test Workbench should wait for the control to appear or to disappear.

   The default timeout is 0, which means Silk Test Workbench does not wait for the control to appear or to disappear.

e) Click **Next**.

9. If you have selected **A variable defined in this visual test**, the **Define a variable-based condition** page opens.

a) Select a pre-defined local variable from the **Select the variable** list.

   For a variable to appear in this list, you must include the local variable in your visual test. To add a new local variable to the visual test, click **New**.

   The **Evaluate as** field, displays the data type of the selected variable.

b) Select the comparison operator to apply to the variable from the **Select the condition** list.

c) In the **Expected value** field, specify the value that the variable is expected to have.

d) Click **Next**.

10. If you have selected **The contents of a Rumba screen**, identify the control on the Rumba screen.

a) Expand the **Identify a control** menu and select one of the following:

- **Application Under Test** – Click this button to identify a visible control directly from the application under test.
- **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
- **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.

b) Click **Next**.

11. If you have selected **A verification asset**, the **Build the Verification Asset** page opens.

a) Click **Create a new verification** to open the **Image Verification** UI, in which you can create a new verification asset.

b) Click **Insert an existing verification** to open the **Browse for Verification** dialog box, in which you can choose the verification asset that you want to execute in your test.

c) *Optional:* Check the **If the verification fails, capture a screenshot** check box to add a screenshot to the result file when the verification fails.

d) Click **Next**.

12. If you have selected **The time taken to run a sequence of steps**, the **Define a timing-based condition** page opens. Specify the sequence of test steps and the range of time within to play back the sequence of test steps.

a) Click on the link to the right of the **Do step** statement to specify the first step in the sequence of steps to verify.

b) Click on the link to the right of the **To step** statement to specify the last step in the sequence of steps to verify.

c) By adapting the values in the two **Should be between** fields, sets the range of time in which the sequence of steps should play back.

   For example, if the range of time is set to 10 to 30 seconds and the actual time to play back a specified sequence of steps is less than 10 seconds or greater than 30 seconds, then a fail message is sent to the result of the visual test. Conversely, if the actual playback time falls between 10 and 30 seconds, then a pass message is sent to the result of the visual test.

d) Select a timer from the **Using timer number** list.

   When timing multiple conditions, you can differentiate the timers by assigning a unique number to each timer. After creating verification logic using a timer, two test steps appear in the **Test Steps** window. The first is labeled `Start timer number X` (with *X* being the specific timer number) and appears just before the first test step in the sequence of steps to verify. The second is labeled `Stop timer X` and appears just after the last step in the sequence of steps to verify.

Tip: If you want to change the sequence of steps to verify, cut and paste either timer test step to the desired location in the **Test Steps** window.

e) Click **Next**.

13. The **Build the Verification** page opens. Create pass/fail messages to include in the result of your visual test. Additionally, if the verification fails, you can send a flag to the result and the **Start Screen** of the current user.
    a) *Optional:* Click on the condition link to the right of the **If** statement to open the *Condition Designer* from which you can edit the condition.
    b) Type text into the **Display the following pass text in the result detail** field that should appear in the result when the verification passes during playback.

       The text appears in the **Result Detail** column of the verification step in the **Test Steps** pane of the **Details** tab.
    c) Type text into the **Otherwise display the following fail text in the result detail** field that should appear in the result when the verification fails during playback.

       The text appears in the **Result Detail** column of the verification step in the **Test Steps** pane of the **Failed** tab, **Flags** tab, and **Details** tab.
    d) *Optional:* Check the **If the verification fails, create a flag in the result** check box to create a flag when the verification fails.

       The flag appears in the **Result Detail** column of the verification step in the **Test Steps** pane of the **Failed** tab, **Flags** tab, and **Details** tab. By default, this flag is automatically assigned to the current user and displays in the **Flags** pane of the **Start Screen** of the current user. If you want to assign this flag to someone else, you must reassign it after playback from the result.
    e) *Optional:* Type a description of the flag when the verification fails into the **Display the following flag description** field.
    f) *Optional:* Check the **If the verification fails, capture a screenshot** check box to add a screenshot to the result file when the verification fails.
14. Click **Next**. The **Summary** page displays the test logic that you have defined. Uncheck the check box at the bottom of the page to prevent the **Summary** page from appearing on subsequent uses of the **Test Logic Designer** wizard. Checking or un-checking this check box updates the **Show Test Logic Designer summary screen** general option. For more information, see *Modifying General Options*.
15. Click **Finish** to insert the test logic into the test steps of your visual test. To modify the test logic, click **Back** and make the desired changes. The test logic is added to the visual test.

Editing Verification Logic Test Steps

1. Open the visual test that contains the test step that you want to edit.
2. In the **Test Steps** window of the visual test, select the verification logic test step that you want to edit. The properties and associated property values for the selected test step appear listed by property category in the **Properties** pane.
3. Edit the available condition properties, verification result text properties, flag settings properties, or miscellaneous properties.

Error Handling Logic

Error handling logic eliminates the need for debugging playback errors by telling the visual test to detect errors, and how to respond to them when they occur. Error handling enables the visual test to playback to completion and provides insight into errors encountered during playback.

Error handling logic consists of the following elements:

```
If <error=TRUE> Then
  __<Action to take>
```

For example, if an error occurs, a fail message and flag is sent, unless you have error handling set to resume the next step:

```
On error 'Resume next'
```

You can create error handling logic using either the **Test Logic Designer** wizard, the Logic Toolbox, or by choosing **Insert** > **Error Handling**.

Adding Error Handling Logic to a Visual Test

This topic describes how you can use the **Test Logic Designer** to add error handling logic to a visual test.

1. Open the visual test.
2. Right-click on the test steps to which you want to add error handling logic.
3. Choose **Insert** > **Test Logic** > **Error Handling**. The **Test Logic Designer** opens.
4. Click **Next**. The **Select a Logic Type** page opens.
5. Click **Next**. The **Build the Error Handler** page opens.
6. Select one of the following items from the list to specify the error-handler logic to use when an error occurs.

Show default playback error dialog	Displays the **Playback Error** dialog box when a playback error occurs, which is used to determine a playback action to take when an error occurs. Use the **Playback Error** dialog box to enter debug mode to diagnose errors in a visual test.
End playback of current visual test	Ends visual test playback when an error occurs. No other steps in the visual test play back. When playback ends, the **Playback Complete** dialog box appears, which determines the action to take when playback completes.
Retry playback of the step on which the error occurred	Plays back the visual test specified in the **Asset** property, then attempts to execute the step that caused the error. If no visual test is specified in the **Asset** property, then playback simply attempts to re-execute the step that caused the error. You can specify the number of replay attempts in the **Number of retries** property. The default number is 5. If the step execution is not successful when the specified number of retries is reached, a playback error is generated.
Resume playback at the next step	Plays back the visual test specified in the **Asset** property, then attempts to execute the step immediately after the step that caused the error. The step that caused the error is not re-executed.
Go to a label	Plays back the visual test specified in the **Asset** property, then navigates to the predefined Label step specified in the **Label** property and continues playback. Using **Go to** allows that visual test to skip steps that could potentially contain failures similar to the one that occurred during playback, and navigate to another section of the test.
First playback another visual test	Plays back a different visual test, which you specify, and then you specify what action happens next. To determine what occurs after the alternate test runs, choose to end the test, retry the step where the error occurred, go to a label, or resume the original test where the error occurred.

7. Click **Next**. The **Summary** page displays the test logic that you have defined. Uncheck the check box at the bottom of the page to prevent the **Summary** page from appearing on subsequent uses of the **Test Logic Designer** wizard. Checking or un-checking this check box updates the **Show Test Logic Designer summary screen** general option. For more information, see *Modifying General Options*.
8. Click **Finish** to insert the test logic into the test steps of your visual test. To modify the test logic, click **Back** and make the desired changes. The test logic is added to the visual test.

Editing Error Handling Logic Test Steps

1. Open the visual test that contains the test step that you want to edit.
2. In the **Test Steps** pane of the visual test, select the error handling logic test step that you want to edit. The properties and associated property values for the selected test step appear listed by property category in the **Properties** pane.
3. Edit the properties as needed.

Test Logic Designer

The **Test Logic Designer** wizard helps you create and insert test logic into your visual test. It is available from the **Insert** > **Test Logic** menu when you have a visual test open. You can also access the wizard through the Logic Toolbox, from which you can learn about the various types of test logic, and then quickly access the appropriate page of the wizard require to create the test logic.

Use the **Test Logic Designer** wizard to insert the following types of test logic:

- Decision Logic
- Repetition Logic
- Verification Logic
- Error Handling Logic

The first page of the **Test Logic Designer** wizard is the **Welcome** page, which displays the list of logic steps in the wizard and the specific type of test logic to design.

To bypass the **Welcome** page on subsequent uses of the wizard and all other Silk Test Workbench wizards, check the check box at the bottom of the page. Checking this check box updates the **Show Wizard Welcome Screens** option in **General** options.

You can resize the wizard by dragging the lower-right corner of any page to the desired height and width. You can also show or hide the **Logic Steps** pane of the wizard by clicking the arrow button next to the **Logic Steps** pane title or by setting the **Show Test Logic Designer steps** pane option in **General** options.

The last page of the **Test Logic Designer** wizard is the **Summary** page. Select the check box at the bottom of the page to prevent the **Summary** page from appearing on subsequent uses of the **Test Logic Designer** wizard. Selecting or clearing this check box updates the **Show Test Logic Designer summary screen** general option. For more information, see *Modifying General Options*.

Selecting Steps

Use the **Select Steps** dialog box to select a single step or a sequence of steps in your visual test.

1. Select a step.
2. *Optional*: To select a sequence of steps, press and hold the `Shift` key, and then select the last step in the sequence.
3. Click **OK**.

Logic Toolbox

The Logic Toolbox is a navigational aide that allows you to quickly insert test logic into your visual test. From the Logic Toolbox, you can learn about the various types of test logic, and then select the desired test logic to insert into a visual test. In most cases, after selecting a type of test logic from the Logic Toolbox, Silk Test Workbench opens to the appropriate page of the **Test Logic Designer** wizard from which you can create and insert the test logic. When you use **Wait for objects to appear or disappear during playback of the visual test**, specify the object properties in the **Properties** pane rather than using the wizard.

The Logic Toolbox organizes common types of test logic into the following categories:

- **Screen Content Logic** – Decision (If) logic or repetition logic that runs a sequence of steps based on the screen controls in your test application.

- **User-defined Follow-up Logic** – Verification logic that sends a user-defined pass/fail message and optional flag to the results of your visual test. Additionally, you can send a flag to the Start Screen of a specified user to notify them about the results of a specific test step or visual test.
- **Error Handling and Synchronization** – Synchronization logic that wait for objects to appear or disappear before performing steps. Error handling logic that performs a user-defined action in the event of an error during playback.

In each category of the Logic Toolbox, you can read a description of each logic type, and then double-click the logic type to quickly insert it into your visual test. For more information, see *Inserting Test Logic Using the Logic Toolbox*.

Inserting Test Logic Using the Logic Toolbox

You can use the Logic Toolbox to insert test logic into your visual test.

1. Open a visual test.

   By default, the **Logic Toolbox** tab appears to the right of the **Test Steps** pane. If this tab does not appear by default, choose **View** > **Logic Toolbox**.
2. Click the **Logic Toolbox** tab to open the Logic Toolbox.

   Tip: To resize the Logic Toolbox, move your pointer to the edge and drag the toolbox to the desired width.
3. Click a logic category to display the different logic types in each category that you can insert into your visual test.

   To view a description of a logic type, click the desired logic type. The description appears at the bottom of the Logic Toolbox.
4. Double-click a logic type to automatically open the appropriate page of the **Test Logic Designer** wizard.

   Or, you can drag and drop the logic type to the appropriate position in the **Test Steps** pane.
5. Use the **Test Logic Designer** wizard to create and insert test logic into your visual test.

Inserting Test Logic During Recording

You can insert test logic during the recording of a visual test.

1. Begin recording.
2. When you reach the appropriate point to add the test logic, perform one of the following steps:

   - Press `Alt+F9`.

     Using this shortcut key combination, temporarily suspends recording and selects the currently highlighted object for verification. The **Test Logic Designer** wizard opens with the selected object set as the control to verify.
   - On the **Recording** dialog box, click **Add Verification** or press `Ctrl+Alt`.

     This option temporarily suspends recording and displays the **Test Logic Designer** wizard. Select the object that you want to verify after the wizard opens.
3. Follow the wizard through the process and click **Finish** to close the wizard and continue recording. The new test logic displays in the **Test Steps** pane of the **Visual Navigator** when recording has completed.

Inserting Test Logic in an Existing Visual Test

You can insert test logic after you create a visual test.

1. Open a visual test.
2. Select the test step or sequence of test steps to include in your test logic.
3. Choose **Insert** > **Test Logic** and then choose the appropriate logic type.

To add verification logic to the visual test, you can also press **Ctrl+Alt**.

The **Test Logic Designer** wizard opens.

4. Follow the wizard, and then click **Finish**.

   For test step descriptions that are too long to view in the **Test Steps** pane, move your pointer over the test step to display a ToolTip containing the entire description.

   The test logic test steps appear in your visual test.

Condition Designer

The **Condition Designer** dialog box helps you quickly define conditions that drive the outcome of decision, repetition, and verification test logic. Access the **Condition Designer** from the **Test Logic Designer** wizard or Logic Toolbox. After creating certain types of test logic, you can also access the **Condition Designer** from the **Properties** pane and edit the condition.

A condition defined using the **Condition Designer** consists of the following elements:

```
<Value 1> <Comparison Operator> <Value 2> = <Return Value>
```

For example:

```
5 > 4 = TRUE
```

The elements of a condition are defined as follows:

- Value 1 and value 2 – The values of the condition serve as the input parameters that are compared against each other. Values in a condition can be a property of a control, a variable, an ActiveData asset, or a literal value.
- Comparison operator – The comparison operator defines the comparison logic of the condition and how the two values in the condition relate to each other. Comparison operators are as follows:
  - Is Equal to (==).
  - Is Not Equal to (! =).
  - Is Greater than (>).
  - Is Less than (<).
  - Is Greater than or Equal to (> =).
  - Is Less than or Equal to (< =).
  - Contains: True if the text in **Value 1** contains the text in **Value 2**. Else false.
  - Does Not Contain: True if the text in **Value 1** does not contain the text in **Value 2**. Else false.
- Return value – When the two values are compared using the comparison operator, the return value of the condition is evaluated to either TRUE or FALSE.

Each type of test logic uses the return value of the condition to determine the resulting action to take. For decision and repetition logic, the action is to run a sequence of test steps. For verification logic, the action is to send a pass/fail message to the result of a visual test or, optionally, flag a specified test step.

Decision Test Logic Example

```
If counterVar Is Equal to 12
Then
Do step 2 to step 4
End If
```

Repetition Test Logic Example

```
Repeat While counterVar Is Equal to 12
Do step 2
To step 4
End Repeat
```

Verification Test Logic Example

```
If counterVar Is Equal to 12
Then
send pass message
else
send fail message
End If
```

Additionally, you can create compound conditions using the logical operators AND or OR. For example, you can create the following compound decision logic condition:

```
If counterVar Is Equal to 12
AND
textVar Is Equal to "Admin"
Then
Do step 2 to step 4
End If
```

Defining a Condition Using the Condition Designer

Use the **Condition Designer** dialog box to define or edit advanced test logic conditions.

1. Access the **Condition Designer** from the **Test Logic Designer** wizard.

   Additionally, you can insert an `If` test step, select the **Condition** property in the **Properties** window, and then click **Condition Designer**.

   The **Condition Designer** opens.
2. In the **Value 1** text box of the **Define a condition** section, specify the first value of the condition, which can be the property of a control, any type of variable, an ActiveData asset, a literal value, or whether a control exists.

   You can enter the value directly in the text box or click **...** to access the **Select the Condition** dialog box.

   Tip: After selecting a value, Silk Test Workbench displays the value type under the **Value 1** text box.

   Note: Data types can be converted to other types, such as a Double converted to a Long, Text, or Boolean. However, it is important to understand how these conversions may affect your data. For more information converting data types, see *Using Expression Designer Functions*.
3. From the **Operator** list, select the comparison operator.

   The comparison operator defines the comparison logic of the condition and how the two values in the condition relate to each other. The following operators are available:

   - Is Equal to (==).
   - Is Not Equal to (! =).
   - Is Greater than (>).
   - Is Less than (<).
   - Is Greater than or Equal to (> =).
   - Is Less than or Equal to (< =).
   - Contains: True if the text in **Value 1** contains the text in **Value 2**. Else false.
   - Does Not Contain: True if the text in **Value 1** does not contain the text in **Value 2**. Else false.
4. In the **Value 2** text box, specify the second value of the condition by following the same process used in step two to specify the first value.
5. From the **Evaluate as** list, select the data type for the outcome of the evaluation of the condition.

   The available data types include:

Text	The variable's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation. **Note:** The text data type is case sensitive, which means the cases of the individual characters in the text string must match during comparison. If the cases do not match, the comparison fails.
Text (case insensitive)	The same as text, except that, for a string comparison during the evaluation, the cases of the individual characters of the string do not have to match.
Number (Long)	The variable's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Number (Double)	The variable's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Boolean (True/ False)	The variable's value is either `True` or `False`.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Enumeration	This variable groups together a set of values and orders them sequentially from 1 to *n*. You declare an enumerated type when you want a variable to hold only a limited number of distinct values. Use this variable type with properties or variables that expect an enumeration data type. An enumeration stores the numeric value in a 64-bit integer in the range -9223372036854775808 to 9223372036854775807. However, the engine only supports 32-bit enumerations, while the UI supports 64-bit enumerations (in future releases, the engine will support 64-bit enumerations).

6. Click **Add** to add the defined condition to the **Conditions** table.

   Disregard the **Logic** list (AND, OR) when creating a single condition statement. Use the **Logic** list only when creating compound conditions.
7. *Optional:* Click **Move Up** or **Move Down** to move a selected condition up a row or down a row respectively in the **Conditions** table.

   When moving a condition that includes a logical operator (AND, OR) to the first row of the **Conditions** table, the logical operator does not move with the condition. It remains in the second row to preserve the conditional logic structure.
8. Optional: When you create compound conditions, create an additional condition using the previous steps, and then select the type of connector logic from the **Logic** list. Click **Add** to add the condition after the previously created condition.
9. *Optional:* Click **Remove** to remove a selected condition from the **Conditions** table.
10. *Optional:* Click **Preview** to display the defined condition in an easy-to-read text format.

Editing a Condition Using the Condition Designer

1. Select the **Condition** property in the **Properties** pane of a test step in a visual test, and then click **Condition Designer**. The **Condition Designer** opens.
2. To edit the values of an existing condition, select the condition to edit in the **Conditions** table.

   List arrows appear in the **Logic**, **Operator**, and **Evaluate as** columns. To edit a value in one of these columns, select the desired item from the list. In the **Value 1** and **Value 2** columns, double-click in the column cell of the selected condition to open the **Select the Condition** dialog box. From this dialog box, you can edit either value.
3. To edit the order of compound conditions, select the condition to edit the order of, and then click **Move Up** or **Move Down**.

You can move a condition containing a logical operator (AND, OR) to the first row of the **Conditions** table. However, the logical operator does not move with the condition. It remains in the second row to preserve the conditional logic structure.

4. Click **OK**.

Selecting a Value for a Condition

When adding conditional logic to a visual test, use the **Select the Condition** dialog box to select a value or to create a literal value for a condition in the **Condition Designer**.

Select a (Literal) – Select this option to display the following literal option:

- **Literal data** – Enter either a text string, a number, or a boolean (true or false) value, and then click **OK**.

1. In the **Define a condition** area of the **Condition Designer**, click **...** to the right of the value text box that you want to populate.

   The **Select the Condition** dialog box opens.
2. Select the type of the value:
   - Click **Property** to select a property of a control.
   - Click **Variable** to select a variable.
   - Click **ActiveData** to select an active data asset.
   - Click **Literal** to create a new literal value, which can be a string, a number, or a boolean value.
   - Click **Exists** to select a verification of whether a specified control exists.
3. If you have selected **Property**, perform the following actions:
   a) Expand the **Identify a Control** menu and select one of the following:
      - **Application Under Test** – Click this button to identify a visible list or combo box control from your test application.
      - **Screen Preview** – Click this button to identify a list or combo box control directly from the **Screen Preview**, if the application under test is not available.
      - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a list or combo box control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.

   b) After identifying a control, select the desired property from the **Properties** list.
4. If you have selected **Variable**, perform the following actions:
   a) Select the variable type from the **Variable type** list or click **New** to create a new variable.

      The following variable types are available:
      - **Local** – Select this to use a local variable, which is a variable that is typically defined and used within a single visual test, and which usually stores the values of other types of variables, for example global or passed-in variables.
      - **Input Parameters** – Select this to use a parameter that is passed in to the visual test.
      - **Output Parameters** – Select this to use a parameter that is passed out by the visual test.
      - **Reserved** – Select this to use a predefined variable, which stores information about the playback and the results of the visual test.

   b) Select the desired variable from the **Variables** list.
5. If you have selected **ActiveData**, perform the following actions:
   a) Expand the **Select** menu to the right of the **ActiveData asset** text box.
   b) To create a new active data asset, click **New ActiveData**.

      For additional information, see *Creating a New Data File for ActiveData Testing*.
   c) To associate an existing active data asset, click **Associate existing ActiveData**.

      For additional information, see *Mapping Data in an ActiveData File to Data in a Visual Test*.

d) After you have specified the active data asset, select the column that you want to use from the **Columns** list.

6. If you have selected **Literal**, type the string, number, or boolean into the **Literal data** text box.
7. If you have selected **Exists**, perform the following actions:
   a) Expand the **Identify a control** menu and select one of the following:
      - **Application Under Test** – Click this button to identify a visible control directly from the application under test.
      - **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
      - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.
   b) Select **Control exists** to verify that the control exists.
   c) Select **Control does not exist** to verify that the control does not exist.
   d) Type a value into the **Timeout** field to specify how long Silk Test Workbench should wait for the control to appear or to disappear.

      The default timeout is 0, which means Silk Test Workbench does not wait for the control to appear or to disappear.
8. Click **OK**.

Expression Designer

The **Expression Designer** helps you quickly create expressions that enhance the functionality of a visual test. An expression is a combination of local variables, reserved variables, ActiveData assets, operators, and functions that return a string or number in either a local variable or in an ActiveData asset.

For example, to test an application that requires ten different order numbers, you can create an expression that increments the value of a variable and returns a unique order number:

```
OrderNumVar=OrderNumVar+1
```

You can then use this expression in a repeat loop and run the loop multiple times. For each iteration of the loop, the order number increments by one and the expression returns a unique order number stored in the local variable `OrderNumVar`.

You can also include ActiveData fields in an expression by specifying an ActiveData column in an expression, and then setting up repetition logic that repeats a sequence of steps using every value in the ActiveData column. For example, you can create an expression that returns only a specified portion of an ActiveData column value such as the values displayed in the following ADDRESS column:

ADDRESS
Anytown, USA 12345
Anycity, USA 23456
Anyplace, USA 34567
Anywhere, USA 45678

To return only the zip code in the ADDRESS column, you can create an expression that includes the ActiveData column "ADDRESS" and the string function `Right`, which returns a specified number of characters from the right side of a string:

```
localVar = Right (ActiveDataAssetName("ADDRESS"),5)
```

This expression returns the last 5 characters from the right side of the string and stores the return value in `localVar` ("12345" in this example). To iterate through the rows of the ActiveData file and return each of the zip codes in the above column, you can set up repetition logic to repeat the expression step using each of the values in the column.

In addition to the string function `Right`, the **Expression Designer** supports a number of other functions and operations.

Defining an Expression Using the Expression Designer

Use the **Expression Designer** dialog box to create or edit expressions that enhance the functionality of a visual test. An expression is a combination of local variables, reserved variables, ActiveData assets, operators, and functions that return a string, number, or object. For more information, see *Expression Designer*.

1. Open a visual test.
2. From the **Test Steps** pane, select the test step after which you want to insert the expression.
3. From the menu bar, choose **Insert** > **Expression**. The **Expression Designer** opens.
4. From the **Output** list, select a local variable or ActiveData asset in which to store the result of the expression.

   Only local variables and ActiveData assets associated with the visual test are listed. Newly created variables and ActiveData assets are automatically associated with the visual test.

   Note: The **Output** list does not display when you create an expression as the value of a property in the **Properties** pane.
5. *Optional:* Click **Select** next to the **Output** list to select an existing ActiveData asset or to create a new variable or ActiveData asset.

   You can select one of the following:

   - Select **New Local Variable** to create local variables for the visual test using the **Add Variable** dialog box. Any local variables created display in the **Output** list and can be used to store the expression result.
   - Select **New ActiveData** to create a new ActiveData asset and associate it with the visual test.
   - Select **Associate existing ActiveData** to associate an existing ActiveData asset with the visual test using the **Browse for ActiveData** dialog box.

   The column names of ActiveData files that are part of associated ActiveData assets also display in the **Output** list and can be used to store the expression result.
6. Use the **Editor** window to create and edit an expression.

   Expressions may contain literal values, variables, ActiveData columns, operators and functions.

   a) To quickly add a variable to an expression, select the desired variable from the **Select a variable (local or reserved)** list, and then click **Add**.

   b) *Optional:* Click **New** to open the **Add Local Variable** dialog box, from which you can create a new variable and automatically associate the variable with the visual test.

   c) To add an ActiveData field, select the desired ActiveData field from the **Select an ActiveData field** list, and then click **Add as** and select the data type.

      The syntax for an ActiveData field in the **Editor** window is as follows:

      ```
      [Project Name].[ActiveData Name].Data Type("Column Name")
      ```

      The supported data types include:

Text	Data in the selected **ActiveData** column is used in the expression as textual data (`String` data type).
Number (Long)	Data in the selected **ActiveData** column is used in the expression as numeric data (`Long` data type).
Number (Double)	Data in the selected **ActiveData** column is used in the expression as numeric data (`Double` data type).

Boolean	Data in the selected **ActiveData** column is evaluated in the expression as True or False (`Boolean` data type).
Number (Long Long)	Data in the selected **ActiveData** column is used in the expression as a 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Enumeration	Data in the selected **ActiveData** column is used in the expression as a set of values and orders them sequentially from 1 to *n*. This data type is used by some user interfaces which require an enumeration as a parameter. For example SAP SendVKey expects an integer enumeration value for the key to be sent. An enumerated type is declared when you want an expression to hold only a limited number of distinct values. You cannot create custom enumerations for usage in visual tests. An enumeration stores the numeric value in a 64-bit integer in the range -9223372036854775808 to 9223372036854775807. However, the engine only supports 32-bit enumerations, while the UI supports 64-bit enumerations (in future releases, the engine will support 64-bit enumerations).

d) You can create a new ActiveData asset or select an existing ActiveData asset by clicking **Select** and choosing the desired option. The new ActiveData asset is automatically associated with the visual test.

7. Click **Test** to assess the validity of the expression. The results of the test display in the **Test results** window.
8. *Optional:* To erase the entire contents of the **Editor** window, click **Clear**.
9. Click **OK**.

Editing an Expression Using the Expression Designer

Use the **Expression Designer** dialog box to create or edit expressions that enhance the functionality of a visual test. An expression is a combination of local variables, reserved variables, ActiveData assets, operators, and functions that return a string, number, or object. For more information, see *Expression Designer*.

1. Open the visual test that contains the expression that you want to change.
2. Select the `Expression` step in the **Test Steps** pane.
3. Select the **Expression** property in the **Properties** pane.
4. Click **Expression Designer**. The **Expression Designer** opens.
5. From the **Output** list, select a local variable or ActiveData asset in which to store the result of the expression.

   Only local variables and ActiveData assets associated with the visual test are listed. Newly created variables and ActiveData assets are automatically associated with the visual test.

   Note: The **Output** list does not display when you create an expression as the value of a property in the **Properties** pane.
6. *Optional:* Click **Select** next to the **Output** list to select an existing ActiveData asset or to create a new variable or ActiveData asset.

   You can select one of the following:

   - Select **New Local Variable** to create local variables for the visual test using the **Add Variable** dialog box. Any local variables created display in the **Output** list and can be used to store the expression result.
   - Select **New ActiveData** to create a new ActiveData asset and associate it with the visual test.
   - Select **Associate existing ActiveData** to associate an existing ActiveData asset with the visual test using the **Browse for ActiveData** dialog box.

The column names of ActiveData files that are part of associated ActiveData assets also display in the **Output** list and can be used to store the expression result.

7. Use the **Editor** window to create and edit an expression.

   Expressions may contain literal values, variables, ActiveData columns, operators and functions.

   a) To quickly add a variable to an expression, select the desired variable from the **Select a variable (local or reserved)** list, and then click **Add**.

   b) *Optional:* Click **New** to open the **Add Local Variable** dialog box, from which you can create a new variable and automatically associate the variable with the visual test.

   c) To add an ActiveData field, select the desired ActiveData field from the **Select an ActiveData field** list, and then click **Add as** and select the data type.

      The syntax for an ActiveData field in the **Editor** window is as follows:

      ```
      [Project Name].[ActiveData Name].Data Type("Column Name")
      ```

      The supported data types include:

Text	Data in the selected **ActiveData** column is used in the expression as textual data (`String` data type).
Number (Long)	Data in the selected **ActiveData** column is used in the expression as numeric data (`Long` data type).
Number (Double)	Data in the selected **ActiveData** column is used in the expression as numeric data (`Double` data type).
Boolean	Data in the selected **ActiveData** column is evaluated in the expression as True or False (`Boolean` data type).
Number (Long Long)	Data in the selected **ActiveData** column is used in the expression as a 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Enumeration	Data in the selected **ActiveData** column is used in the expression as a set of values and orders them sequentially from 1 to *n*. This data type is used by some user interfaces which require an enumeration as a parameter. For example SAP SendVKey expects an integer enumeration value for the key to be sent. An enumerated type is declared when you want an expression to hold only a limited number of distinct values. You cannot create custom enumerations for usage in visual tests. An enumeration stores the numeric value in a 64-bit integer in the range -9223372036854775808 to 9223372036854775807. However, the engine only supports 32-bit enumerations, while the UI supports 64-bit enumerations (in future releases, the engine will support 64-bit enumerations).

   d) You can create a new ActiveData asset or select an existing ActiveData asset by clicking **Select** and choosing the desired option. The new ActiveData asset is automatically associated with the visual test.

8. Click **Test** to assess the validity of the expression. The results of the test display in the **Test results** window.

9. *Optional:* To erase the entire contents of the **Editor** window, click **Clear**.

10. Click **OK**.

Using Expression Designer Operators

The **Expression Designer** supports the following operators:

Arithmetic Operators

Operator	Operator Name	Syntax	Example
+	Plus Operator	+a	+1
-	Negation Operator (unary)	-a	-1
+	Addition Operator	a + b	1 + 2 = 3
-	Subtraction Operator	a - b	2 - 1 = 1
*	Multiplication Operator	a * b	2 * 2 = 4
/	Division Operator	a / b	4 / 2 = 2
%	Modular Operator	a % b	5%2=1 (remainder)

Comparison Operators

Operator	Operator Name	Syntax	Example
!	Logical Boolean Negation Operator (unary)	!a	! a (not a)
&&	Logical AND	a && b	1 < 5 && 5 < 10 = true
II	Logical OR	a || b	1 < 5 || 5 > 10 = true
=	Assignment Operator	a = b	a = 3
==	Equal To Operator	a == b	3 == 3
!=	Not Equal To Operator	a != b	4 != 3
>	Greater Than Operator	a > b	4 > 3
>=	Greater Than Or Equal To Operator	a >= b	4 >= 3
<	Less Than Operator	a < b	3 < 4
<=	Less Than Or Equal To Operator	a <= b	3 <= 4

Bitwise Operators

Operator	Operator Name	Syntax	Example
~	Bitwise NOT (performs logical negation on an expression by inverting the bits of a binary number)	a = ~ b	~1 = -2 where 00000001 = 1 11111110 = -2
&	Bitwise AND (performs logical conjunction on two expressions)	a = b & c	1 & 1 = True 0 & 1 = False 1 & 0 = False 0 & 0 = False
|	Bitwise OR (performs logical disjunction on two expressions)	a = b | c	1 | 1 = True 0 | 1 = True 1 | 0 = True

Operator	Operator Name	Syntax	Example
^	Bitwise XOR (performs logical exclusion on two expressions)	a = b ^ c	0 | 0 = False 1 ^ 1 = False 0 ^ 1 = True 1 ^ 0 = True 0 ^ 0 = False

Using Expression Designer Functions

The **Expression Designer** supports the following functions:

Mathematical Functions

Function/Description	Syntax	Example
`Abs`: Returns the absolute value of a number.	`Abs`(expression)	`Abs`(-2) and `Abs` (2) both return 2.
`Round`: Returns a number rounded to the nearest integer.	`Round`(expression)	`Round`(1.3333) returns 1
`Max`: Compares two numbers and returns the largest number.	`Max`(number1,number2)	`Max`(-123,12) returns 12
`Min`: Compares two numbers and returns the smallest number.	`Min`(number1,number2)	`Min`(-123,12) returns -123

String Functions

Function/Description	Syntax	Example
`Contains`: Returns whether a string is contained within another or not.	`Contains`(string1,string2)	`Contains("Hello World", "Hello")` returns true. `Contains("Hello World", "Goodbye")` returns false
`Find`: Returns the position of the first occurrence of one string within another.	`Find`(string1,string2)	`Find`("abcd","c") returns 3
`FormatDate`: Returns a date string based on the date, date format, and locale.	`FormatDate`(date, format, locale)	`FormatDate`("May 13, 2009", "MM/dd/yy", "en-US") returns "05/13/09"
`FormatTime`: Returns a time string based on the time, time format, and locale.	`FormatTime`(time, format, locale)	`FormatTime`("23:02:33", "hh:mm tt", "en-US") returns "11:02 PM "
`Left`: Returns a specified number of characters from the left side of a string.	`Left`(string,length)	`Left`("abcd",2) returns "ab"
`Length`: Returns the number of characters in a string.	`Length`(string)	`Length`("abcd") returns 4

Function/Description	Syntax	Example
`MakeLower`: Converts a text string to lowercase characters.	`Makelower`(string)	`Makelower`("ABCD") returns "abcd"
`MakeUpper`: Converts a text string to uppercase characters.	`Makeupper`(string)	`Makeupper`("abcd") returns "ABCD"
`Mid`: Returns a specified number of characters from the middle of a string.	`Mid`(string,start,length)	`Mid`("abcd",2,2) returns "bc"
`Replace`: Returns a text string in which a specified substring has been replaced with another substring a specified number of times.	`Replace`(string, substring to replace, replacement substring)	`Replace`("abc def","bc","xy") returns "axy def"
`Right`: Returns a specified number of characters from the right side of a string.	`Right`(string,length)	`Right`("abcd",2) returns "cd"
`Trim`:Returns a specified string without leading spaces or trailing spaces.	`Trim`(string)	`Trim`(" abcd ") returns "abcd"
`TrimLeft`: Returns a specified string without leading spaces.	`TrimLeft`(string)	`TrimLeft`(" abcd ") returns "abcd "
`TrimRight`: Returns a specified string without trailing spaces.	`TrimRight`(string)	`TrimRight`(" abcd ") returns " abcd"

Conversion Functions

Function/Return Type	Syntax	Range for Expression
`ToBool`: Boolean	`ToBool`(expression)	Any valid string or numeric expression.
`ToDouble`: Double	`ToDouble`(expression)	1.7E +/- 308
`ToLong`: Long	`ToLong`(expression)	-2,147,483,648 to 2,147,483,647; fractions are rounded.
`ToString`: Text	`ToString`(expression)	Returns for `ToString` depend on the content of the expression.
`ToLongLong`: Integer	`ToLongLong`	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`ToEnumeration`: Integer	`ToEnumeration`	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807  **Note:** Because the Open Agent does not support 64-bit data types, the recommended range for this expression is -2,147,483,648 to 2,147,483,647.

Conversion Rules

When converting data types, observe the following rules:

Converting a Double to a Long	Decimals are truncated. Numbers outside the bounds of a Long are wrapped. For example, the Double number 49.6 is rounded down (i.e., .6 is truncated) to 49. The Double number -2147483649 is wrapped to the highest limit of a Long, which is +214748367.
Converting a String to a Double or Long	Conversion stops at the first unrecognized character, such as an alpha character. For a Long, decimals are truncated and numbers outside the bounds of a Long are wrapped. For example, "123.7 oranges" converts to Double "123.7" and Long "123".
Converting to a Boolean	Any non-zero numeric character, the text "true", or any text that converts to a number is converted to True. The zero character, the text "false", or any text that cannot convert to a number is converted to False.
Converting a Boolean to a Double or Long	`True = 1` and `False = 0`.

FormatDate Function

Description

An Expression Designer function that returns a string expression based on the date, date format, and locale specified by the user.

Syntax

```
local variable = FormatDate (Date, Format, Locale)
```

The `FormatDate` function syntax has the following parameters:

Date A date string or a variable containing a date string that is valid for the specified locale. To specify the current system date, use "Current". To specify a date string, enter the desired date string in quotes ("January 5th, 2009"). For variables containing a date string, use the variable name without quotes.

Format A date format that consists of a combination of the following elements that are valid for the specified locale:

Day

- `d` – Day of month as digits with no leading zero for single-digit days ("1"-"9", "10"-"31").
- `dd` – Day of month as digits with leading zero for single-digit days ("01"-"09", "10"-"31").
- `ddd` – Day of week as a three-letter abbreviation (Mon, Tue, Wed, and so on, in English).
- `dddd` – Day of week as its full name (Monday, Tuesday, Wednesday, and so on, in English).

Month

- `M` – Month as digits with no leading zero for single-digit months ("1"-"9", "10"-"12").
- `MM` – Month as digits with leading zero for single-digit months ("01"-"09", "10"-"12").
- `MMM` – Month as a three-letter abbreviation (Jan, Feb, Oct, and so on, in English).
- `MMMM` – Month as its full name.

Year

- y – Year as last two digits, but with no leading zero for years less than 10 ("0"-"9", "10"-"99").

- yy – Year as last two digits, but with leading zero for years less than 10 ("00"-"09", "10"-"99").
- yyy – Year as last two digits, but with leading zero for years less than 10 ("00"-"09", "10"-"99").
- yyyy – Year represented by full four digits.

Era gg – The year-division, or era designation (Japanese Era Name calendar).

Note: Silk Test Workbench verifies if an era exists for the date and locale specified. If it exists, Silk Test Workbench uses the era for the default calendar. If it does not exist, Silk Test Workbench determines whether an era exists for the alternate calendar in the same locale. If an alternate calendar exists, Silk Test Workbench formats the entire string in the alternate calendar; otherwise, Silk Test Workbench formats the string in the default calendar and ignores the era format (gg).

Locale In general terms, a locale defines the user's language, country and any special variant preferences, such as the formats used for dates and times, that the user wants to see in their user interface.

The `Locale` parameter applies a specified locale definition for how a date string is interpreted and displayed in the output of the `FormatDate` function. Values for this parameter are based on Windows operating system locale names. An example of a locale name for French (Canada) is "fr-CA". For a list of supported locales, refer to the "Locale Identifier Constants and Strings" topic in Microsoft's National Language Support Reference.

Note: The availability of a locale is based on your operating system version and Regional Options settings.

Remarks

All parameters are required. To use the default value of the `Format` or `Locale` parameter, enter "Default" in the expression. In the following example, the `FormatDate` function returns the current system date using the system date format and locale of the operating system:

```
FormatDate("Current", "Default", "Default")
```

The following examples show how to use the elements of the `Format` parameter to customize the output of the `FormatDate` function:

```
FormatDate("Jan 9, 2009", "dddd, MMM-dd-yyyy",
 "Default") returns "Friday, Jan-09-2009"

FormatDate("Jan 9, 2009", "d/M/yy", "Default")
 returns "9/1/09"

FormatDate("Jan 9, 2009", "yy", "Default")
 returns "09"
```

The following example is run on an English operating system and shows how to use the `Locale` parameter to convert a French locale date to a Spanish locale date. This is a two step process in which you convert a non-numeric French date (2009 janvier 9) to a numeric French date (2009-01-09) format. In the second step, you convert the numeric date to a Spanish date (2009-enero-09):

```
FormatDate("2009 janvier 9", "yyyy-MM-dd",
 "fr-FR") returns "2009-01-09"

FormatDate(FrNumericDate, "yyyy-MMMM-dd", "es-ES")
 returns "2009-enero-09"
```

FormatTime Function

Description

An Expression Designer function that returns a string expression based on the time, time format, and locale specified by the user.

Syntax

```
local variable = FormatTime(Time, Format, Locale)
```

The `FormatTime` function syntax has the following parameters:

Time A time string or a variable containing a time string that is valid for the specified locale. To specify the current system time, use "Current". To specify a time string, enter the desired time string in quotes ("01:12:23"). For variables containing a time string, use the variable name without quotes.

Format A time format that consists of a combination of the following elements that are valid for the specified locale:

Hours

- `h` – Hours with no leading zero for single-digit hours; 12-hour clock
- `H` – Hours with no leading zero for single-digit hours; 24-hour clock
- `hh` – Hours with leading zero for single-digit hours; 12-hour clock
- `HH` – Hours with leading zero for single-digit hours; 24-hour clock

Minutes

- `m` – Minutes with no leading zero for single-digit minutes
- `mm` – Minutes with leading zero for single-digit minutes

Seconds

- `s` – Seconds with no leading zero for single-digit seconds
- `ss` – Seconds with leading zero for single-digit seconds

Time Marker

- `t` – One character time marker string, such as A or P.
- `tt` – Multicharacter time marker string, such as AM or PM.

Locale In general terms, a `Locale` defines the user's language, country and any special variant preferences, such as the formats used for dates and times, that the user wants to see in their user interface.

The `Locale` parameter applies a specified locale definition for how a time string is interpreted and displayed in the output of the `FormatTime` function. Values for this parameter are based on Windows operating system locale names. An example of a locale name for French (Canada) is "fr-CA". For a list of supported locales, refer to the "Locale Identifier Constants and Strings" topic in Microsoft's National Language Support Reference.

Note: The availability of a locale is based on your operating system version and Regional Options settings.

Remarks

All parameters are required. To use the default value of the `Format` or `Locale` parameter, enter "Default" in the expression. In the following example, the `FormatTime` function returns the current system time using the system time format and locale of the operating system:

```
FormatTime ("Current","Default", "Default")
```

The following examples show how to use the elements of the `Format` parameter to customize the output of the `FormatTime` function:

```
FormatTime("9:05:08 PM", "hh:mm:ss tt", "en-US") returns "09:05:08 PM"
```

```
FormatTime("9:05:08 PM", "HH:mm:ss", "en-US") returns "21:05:08"

FormatTime("9:05:08 PM", "tt", "Default") returns "PM"
```

The following example shows how to use the `Locale` parameter to affect the output for English (United States) and English (United Kingdom) locales:

```
FormatTime("14:30", "Default", "en-US") returns "2:30:00 PM"

FormatTime("14:30", "Default", "en-GB") returns "14:30:00"
```

Working with Automation Steps

This section lists the topics that explain how to use automation steps within visual tests.

Creating Automation Test Steps Without Recording

Automation test steps are steps that perform actions against controls in a test application in order to test the application. For example, steps that select an item, enter text, or click a control in a test application are automation steps.

Automation steps are usually recorded. When you interact with controls in a test application while recording a visual test, Silk Test Workbench typically records each interaction (such as a click) as a separate step.

You can, however, create automation test steps and insert them into a visual test without recording. Creating automation steps without recording lets you accommodate slight changes in a test plan without having to re-record an entire test. To create an automation test step, you can use the **Identify Object** dialog box and interact with the test application or a captured screen image in the **Screen Preview**. Using the **Identify Object** dialog box lets you see how Silk Test Workbench recognizes the various controls in the test application, which can provide insight into how to optimize the testing of the application.

Creating a Control Automation Step Using a Test Application

When working with a test application, you can create individual steps that perform test actions. Creating individual steps means you do not have to re-record the visual test. Simply create additional actions against the test application as needed.

You can also create additional actions against controls using captured screens in the **Screen Preview**.

1. Open the visual test in which you want to identify a control.
2. Open the test application that contains the control.
3. Select the step preceding the insertion point.

   The step you select is where you want to perform an action against the control.
4. Choose **Insert** > **Control From** > **Application Under Test**. The **Recording** dialog box opens.
5. Select the control that you want to use in the test application. The **Select Action** dialog box opens.
6. Determine the action to use on the control. The automation step appears below the selected step. Properties for the step appear in the **Properties** pane. The selected action appears as the **Method** property in the **Action** category. You can edit properties for the step to further control how the action executes.

Creating a Control Automation Step Using the Screen Preview

Before you create a control automation step using the screen preview, ensure that the option **Control capture** is set to `Yes`.

When working with a test application, you can create individual steps that perform test actions. Creating individual steps means you do not have to re-record the visual test. Create additional actions against test application screens that have been captured in the **Screen Preview** as needed.

You can also create additional actions against controls using the test application.

1. Open the visual test that contains the captured screen from which you want to identify a control.
2. Select a step that displays the test application screen or control in the **Screen Preview**. To do this, click the step in the Test Steps pane or click a thumbnail in the Storyboard.

   To better see individual controls in the **Screen Preview**, choose **Actions** > **Zoom** in the **Screen Preview** and choose a zoom percentage.
3. Display the desired control in the **Screen Preview** and choose **Actions** > **Insert Control From** > **Screen Preview**.
4. Identify the control in the **Screen Preview**. The **Select Action** dialog box opens.
5. Determine the action to use on the control. The automation step appears below the selected step. Properties for the step appear in the **Properties** pane. The selected action appears as the **Method** property in the **Action** category. You can edit properties for the step to further control how the action executes.
6. Click **OK**.

Creating a Control Automation Step Using the Identify Object Dialog

When working with a test application, you can create individual steps that perform test actions. Creating individual steps means you do not have to re-record the visual test. Simply create additional actions against the test application as needed.

You can also create additional actions against non-visible controls using the **Identify Object** dialog box.

1. Open the visual test in which you want to identify a control.
2. Open the test application that contains the control.
3. Select the step that is preceding the insertion point.

   The step you select is where you want to perform an action against the control.
4. Choose **Insert** > **Control From** > **Identify Object Dialog**.
5. If you are testing a web application, the **Select Browser** dialog box opens. Select the browser on which you want to identify the control and click **Start Identify**. The **Identify Object** dialog box opens.
6. Click **Start Identify**.
7. Move the cursor to the application that you are testing. Controls appear highlighted as the cursor passes over them in the application. The related locator string or object map item shows in the **Selected Locator** text box.
8. Click the control that you want to use as it is highlighted.
9. *Optional:* Click **Show Locator Details** to display the locator tree and any related attributes in the **Locator Attribute** table.
10. *Optional:* You can replace a recorded locator attribute with another locator attribute from the **Locator Attribute** table.

    For example, your recorded locator might look like the following:

    ```
    /BrowserApplication//BrowserWindow//input[@id='loginButton']
    ```

    If you have a `textContents Login` listed in the **Locator Attribute** table, you can manually change the locator to the following:

    ```
    /BrowserApplication//BrowserWindow//input[@textContents='Login']
    ```

    The new locator displays in the **Selected Locator** text box.
11. Click **Test** to verify that the locator recognizes the correct control.
12. To add a new step that uses the locator to the visual test, click **Paste**. The **Select Action** dialog box opens. Select the action that should be performed against the control and click **OK**.

Using Control Property Data in a Visual Test

Silk Test Workbench lets you use property values of any control in a test application as data in a visual test. To do this, you set the property variable to a local variable that has been defined in the visual test. For

example, you can set the ID of a control to a variable, then use the variable in test logic to ensure no other control in the application uses the same ID.

The properties that can be set to a variable depend on the properties available for a control. Not all controls have the same set of available properties.

To set a control property value to a variable, the variable must be created for the visual test.

1. Open the visual test in which you want to set a control property to a variable.
2. Create a local variable to hold the value of the control property.
3. Select the step that precedes where you want to insert the step to set the control property to a variable.

   Make sure the application containing the control with the desired property data will be available at the point in the test where the step is inserted.
4. Choose **Insert** > **Property from Control**. The step is created after the selected step. Properties for the step appear in the **Properties** pane.
5. In the **Properties** pane, in the **Locator** text box, identify the control to which you want to assign a variable.
   - **Application Under Test** – Click this button to identify a visible list or combo box control from your test application.
   - **Screen Preview** – Click this button to identify a list or combo box control directly from the **Screen Preview**, if the application under test is not available.
   - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a list or combo box control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.
   - **Select** – Click this button to assign a literal, variable, expression result, or ActiveData value.
6. From the **Property** list, specify whether to use a literal, variable, expression, or ActiveData as the property of the control.

   This value is stored in the local variable.
7. From the **Local variable name** list, select the variable to use with the control and property that you selected.
8. *Optional*: Edit the **Miscellaneous** category properties to describe general information about the step.

Screen Preview and Step Synchronization in Visual Tests

During record and playback, Silk Test Workbench captures application snapshots that include captured screens and enumerations of various controls with various properties. The captured screens display in the **Screen Preview**.

Silk Test Workbench takes snapshots under the following circumstances:

- Before every automation test step during recording.

  Note: For SAP applications, snapshots occur when the screen changes rather than before every automation test step.
- When executing a `Using` step in a visual test, a snapshot of the result.
- When a playback error occurs.

Tip: When working with test applications that are slow to refresh the contents of the application screens, you can configure Silk Test Workbench to wait before taking a snapshot of the application screen. For more information, refer to the **Attach to screen shot delay** option.

Automation Test Steps

Every automation test step has its own snapshot and is labeled as `(group screen)` or `(step screen)` in **Screen Preview**. The `(group screen)` contains the main context of the current dialog box or Web page. While the `(step screen)` is associated with a specific action on the current dialog box or Web page.

Silk Test Workbench takes the application snapshots before the actual test step that references the snapshot. For automation steps where the screen context changes, such as a new dialog box or Web page opening, a `Using` step is associated with a screen capture that represents the application state before the automation step. In the **Screen Preview**, the `Using` step is labeled `(group screen)`. For example, a `(group screen)` for a Web page in the **Screen Preview** might correspond with a `Using Browser Application 'BrowserApplication` step in the **Test Steps** pane. The test steps that follow the `Using` step correspond with a single action that occurs on the context of the current dialog box or Web page and are labeled `(step screen)` in the **Screen Preview**. For instance, if you enter a zip code in a text field, the corresponding test step might be `Enter '92109'` and the **Screen Preview** shows a black rectangle around the zip code text field and the zip code value before the value that you entered. So, if the value before `92109` was `18966`, the **Screen Preview** shows `18966`.

If you disable a `Using` step, the steps that follow the `Using` step are associated with the `Using` step prior to the `Using` step that you disabled. For instance, if you disable `Using` step 26, the following test steps use the `Using` step prior to step 26.

Note: If you insert a control from screen preview, the `(group screen)` snapshot displays in **Screen Preview**. Because the control was not recorded, no snapshot was taken. As a result, the `(group screen)` is used.

Other Test Step Types

In most cases, test steps that are not associated with automation steps do not display a snapshot in **Screen Preview**. For instance, if you insert a script into a visual test, **Screen Preview** displays the text that displays in the **Test Steps** pane.

Configuring Settings

By default, Silk Test Workbench records both group and individual screen captures during recording. You can change these settings to only record group screen captures or to record no screen captures using the **Options** dialog box. Choose **Tools** > **Options** expand **Record**, click **Output** and then click **Visual test** to configure these options.

You can configure playback screen capture settings for both group and individual screen captures in a visual test or you can set global options that apply to all visual tests in the **Options** dialog box. Choose **Tools** > **Options** expand **Playback**, click **Results** and then click **Visual test** to configure these options.

To configure individual settings for visual tests, record the visual test and then insert a `Set playback setting` step to override the global playback value. Settings include: `Screen capture` and `Screen capture (test steps)`.

For example, you might set the playback setting for **Screen capture (test steps)** to **No** for a specific step to improve performance and then return the **Screen capture (test steps)** to **Yes** for any remaining steps where an individual screen capture is imperative.

Updating Captured Screens for a Screen Test Step

From the **Screen Preview** of a visual test or result, you can quickly update previously captured screens of a screen test step. Captured images represent the state of the application before its associated step is executed. Silk Test Workbench provides two methods for updating captured screens:

Recapturing screens from the test application	This method is performed from the **Screen Preview** of a visual test and involves opening the test application to the window containing the new screen and recapturing it.
Using screens captured during playback	This method is performed from the **Screen Preview** of a result and does not require accessing the test application.

Updating Captured Screens Using Screens Recorded During Playback

From the **Screen Preview** of a visual test or result, you can quickly update previously captured screens for a screen test step. Captured images represent the state of the application before its associated step is executed.

1. Open the result of a visual test.
2. In the **Test Steps** pane, select the step with the screen requiring updating or click the Storyboard thumbnail containing the screen.
3. To update only the current step, choose **Actions** > **Update Screen** from the **Screen Preview** or the **Test Steps** pane. Silk Test Workbench opens the visual test, displays the updated screen in the **Screen Preview**, and displays a confirmation message box asking if you want to update the screen item in the visual test. For this feature to work, the result and visual test must have the same name. If you rename a visual test, you must rename its result as well. Otherwise Silk Test Workbench does not find the visual test when you attempt to update the screen item in the visual test.

   Note: Before updating a screen test step, you can view the differences between the screen captured during recording and the screen captured during playback, by choosing **Actions** > **Show Differences** > **Side by Side**.
4. Click **Yes** to update the visual test with the screen captured during playback. The visual test opens.
5. To update all screens in the visual test, choose **Actions** > **Update All Screens** from the **Screen Preview** or the **Test Steps** pane. For this feature to work, the result and the visual test must have the same name. If you rename a visual test, you must rename its result as well. Otherwise Silk Test Workbench does not find the visual test when you attempt to update the screen item in the visual test.
6. Check the **Return to Result** check box on the confirmation message box to display the **Result** window after you click **Yes**.

Updating Captured Screens from the Test Application

From the **Screen Preview** of a visual test or result, you can quickly update previously captured screens for a screen test step. Captured images represent the state of the application before its associated step is executed.

1. Start the test application.
2. Navigate the test application to display the window containing the screen whose information requires update in the visual test.
3. With the visual test open, select the screen step with the screen requiring updating or click the Storyboard thumbnail containing the screen.
4. Choose **Actions** > **Update Screen**. Silk Test Workbench highlights the screen in the test application corresponding to the test step being updated if found. The highlight is a rectangular border that flashes around the test screen. If the screen cannot be found in the test application, a message appears indicating that the window in the test application cannot be located. The window may have been renamed, or window properties changed to prevent Silk Test Workbench from recognizing the window. Click **Yes** to re-identify the updated window in the test application, or you can identify another window. Click **No** to return to the visual test without recapturing any window information.
5. Another message displays asking if you want to update the test step with the updated information captured from the test application screen.
   - Click **Yes** to update the test step.
   - Click **No** to close the message box without making any updates to the test step.

Using Item Identifiers in Visual Tests

In a visual test, you can only specify an item identifier as an index, which means inside brackets, and not as a simple number.

Example

For example, to return the text of the cell in the second row and in the second column of a `DomTable` for the `GetCellText` method, add the following step to the visual test:

```
Return cell text from row '[2]', column '[2]' and assign the
return value to st_LastReturnValue
```

In the parameters section of the properties of this step, set the row parameter to `[2]`, and set the col parameter also to `[2]`.

Note: Setting these parameters to 2 instead of `[2]` will lead to an error, as the brackets are required for item identifiers in visual tests.

Reusing Test Data in Visual Tests

Lists the types of reuse that are available to visual tests.

Variables in Visual Tests

Testing applications usually involves using data. You can use data to test a part of an application, or use data throughout that application and other applications. You can type data into fields to test how the field accepts user input. You can also use data to determine how tests playback against a test application.

During recording, data input against a control is logged as literal data. For example, when a WPF TextBox control has a value typed into it, the resulting test step looks similar to the following: `Enter "Ford"`.

This is acceptable, but the visual test can never use data other than the value *Ford* in that test step. However if the value *Ford* is replaced with a variable representing that data, whatever data is used in the variable is input in the TextBox control.

Using variables provides greater testing flexibility because data used in visual tests does not have to be constant.

Defining a variable for a visual test includes defining what kind of data it can hold. This characteristic of a variable is known as a *data type*. For visual tests, you can use variables of the following data types:

- Text (String)
- Number (Long)
- Number (Double)
- Boolean (True/False)
- Number (Long Long)
- Enumeration

For example, when Silk Test Workbench records data input into controls, if the value recorded is in quotes (""), it is usually a `Text` data type and you can replace it with a variable of the `Text` data type.

Visual tests use the following types of variables:

- *Local* – These are variables that are typically defined and used within a single visual test. Local variables can be used to store the values of other types of variables such as global or visual test variables. Choose **Insert** > **Variable** > **Add Local** to add a local variable, or define them in the **Properties** pane for the `<<Start>>` step.
- *Global* – Global variables are defined within the context of a visual test, but can be used in any visual test that executes the visual test containing the global variable. When working with a visual test in the **Visual Navigator**, choose **Insert** > **Variable** > **Set Global** to define the value of a global variable. In other visual tests, choose **Insert** > **Variable** > **Get Global** to assign a global variable's value to a local variable.

- *Parameters* – These are variables passed from one visual test to another visual test.
- *Input/Output Parameters* – Use input parameters to pass data from a visual test to a script. Use output parameters to pass data from a script to a visual test.
- *Reserved* – These are variables that are predefined. They can be used to retrieve information about visual test playback and other useful playback result information. Reserved variable names always begin with the ST prefix. Reserved variables are read-only, but can be used in expressions along with the other variable types.

Tip: Data can also be reused in scripts by using variables. For more information, see *Variables in Scripts*.

Monitoring Variable Use During Visual Test Playback

Silk Test Workbench lets you track the use of variables during visual test playback. Tracking how variables are used and the values they contain during playback can provide valuable information for visual test debugging.

Before playing back a visual test, display the Local Variables window, which indicates the state of variable usage as they are created, used and deleted during visual test playback.

Set breakpoints in a visual test or set one breakpoint and step through test steps during playback with the Local Variables window to get a snapshot of variable use at critical points in the execution of any test.

1. Open the visual test that contains the variables you want to monitor during playback.
2. Choose **Debug** > **Local Variables**. The Local Variables window opens. You can move, dock, hide, or resize the window.
3. Set breakpoints as desired to stop playback at selected points to make variable tracking easier.
4. Playback the visual test. When in debug mode, as playback stops at the selected points, each existing variable and its current value displays in the Local Variables window.

Tip: The Local Variables window also indicates the name of the visual test using the variables. This is useful when playing back multiple visual tests or when one visual test plays back another within its steps.

Tip: To copy a variable from the Local Variables window to the clipboard, right-click on the variable. Select **Copy** to copy everything for the currently selected variable, including any child variables. Otherwise, select **Copy Value** to copy only the **Value** column of the currently selected variable. If the **Value** column of the variable is empty, this menu item is disabled.

Local Variables

Local variables are variables created for use within a visual test. They are defined in the `<<Start>>` step of a visual test, edited in the **Properties** window, and are available to the visual test during playback. Once local variables are created, they are available to the visual test in which they are created, and can be used:

- In test logic
- In results comments
- In expressions
- To hold global variable values set in other visual tests

Adding Local Variables to a Visual Test

Local variables are variables created for use within a visual test. They are defined in the `<<Start>>` step of a visual test, edited in the **Properties** window, and are available to the visual test during playback. Once local variables are created, they are available to the visual test in which they are created.

1. Open the visual test to which you want to add local variables.
2. Choose **Insert** > **Variable** > **Add Local**. The **Add Local Variable** dialog box opens.

3. Type a name for the variable into the **Variable name** field.

   Valid characters for local variable names are uppercase and lowercase alphanumeric characters and the underscore ("ABC", "abc", "_"). Additionally, local variable names must be unique.

   Note: Variable names cannot begin with "st_".

   Silk Test Workbench informs you if a variable name which is being added or edited is a duplicate variable name. If this happens, you must supply a unique name for the variable being added or edited.
4. Select a type for the variable from the **Type** list.

   When adding or editing local variables, the displayed data type is based on the data of the variable being added using the properties window for the `<<Start>>` step. When storing formula results, select the type from the list.

   Silk Test Workbench supports the following variable types:

Text	The variable's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation. **Note:** The text data type is case sensitive, which means the cases of the individual characters in the text string must match during comparison. If the cases do not match, the comparison fails.
Number (Long)	The variable's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Number (Double)	The variable's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Boolean (True/ False)	The variable's value is either `True` or `False`.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Enumeration	This variable groups together a set of values and orders them sequentially from 1 to *n*. You declare an enumerated type when you want a variable to hold only a limited number of distinct values. Use this variable type with properties or variables that expect an enumeration data type. An enumeration stores the numeric value in a 64-bit integer in the range -9223372036854775808 to 9223372036854775807. However, the engine only supports 32-bit enumerations, while the UI supports 64-bit enumerations (in future releases, the engine will support 64-bit enumerations).

5. *Optional:* Type an initial value for the variable into the **Initial value** field. The value can be changed in a visual test or script if passed to a script. The value must be consistent with type. For example, if the type is number (long), the initial value cannot contain characters that are not integers, such as text characters. For boolean variable types, select **False** or **True** from the list. Initial values are not required for variables, so this field can be left blank. The value can be determined elsewhere in a visual test.
6. Click **OK**. The variable's name and initial value (if any) display under the selected variable type in the **Properties** pane for the `<<Start>>` step.
7. Repeat the preceding step to create other local variables as needed. The value area for the variable types indicate the number of variables of the type that are currently defined. For example, if the value area for the `Number(Long)` item is **3 Items**, there are three local variables currently defined for the visual test that are of the `Long` data type.

Editing Local Variables For a Visual Test

You can edit local variables in the **Properties** pane of the `<<Start>>` step.

1. Open the visual test that contains the local variables that you want to edit.
2. Select the `<<Start>>` step of the visual test. Properties for the `<<Start>>` step display in the **Properties** pane.
3. In the **Properties** pane, find the **Variables** category or double-click **Variables** to expand the category and display the local variables defined for the visual test.

   Defined variables in the **Variables** category include the following types:

Visual test	Variables that are defined in another visual test and passed to this visual test.
Text	The variable's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation.  **Note:** The text data type is case sensitive, which means the cases of the individual characters in the text string must match during comparison. If the cases do not match, the comparison fails.
Number (Long)	The variable's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Number (Double)	The variable's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Boolean (True/ False)	The variable's value is either `True` or `False`.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Enumeration	This variable groups together a set of values and orders them sequentially from 1 to *n*. You declare an enumerated type when you want a variable to hold only a limited number of distinct values. Use this variable type with properties or variables that expect an enumeration data type. An enumeration stores the numeric value in a 64-bit integer in the range -9223372036854775808 to 9223372036854775807. However, the engine only supports 32-bit enumerations, while the UI supports 64-bit enumerations (in future releases, the engine will support 64-bit enumerations).

4. Click in the value area of the variable to be edited, then click the **Edit variable** button in the value area. The **Edit Variable** dialog box opens.
5. Type a name for the variable into the **Variable name** field.

   Valid characters for local variable names are uppercase and lowercase alphanumeric characters and the underscore ("ABC", "abc", "_"). Additionally, local variable names must be unique.

    **Note:** Variable names cannot begin with "st_".

   Silk Test Workbench informs you if a variable name which is being added or edited is a duplicate variable name. If this happens, you must supply a unique name for the variable being added or edited.
6. *Optional:* Type an initial value for the variable into the **Initial value** field. The value can be changed in a visual test or script if passed to a script. The value must be consistent with type. For example, if the type is number (long), the initial value cannot contain characters that are not integers, such as text characters. For boolean variable types, select **False** or **True** from the list. Initial values are not required for variables, so this field can be left blank. The value can be determined elsewhere in a visual test.
7. Click **OK**. The variable's name and initial value (if any) display under the selected variable type in the **Properties** pane for the `<<Start>>` step.
8. Repeat the preceding step to edit other local variables as necessary.

   If you are editing variables to pass to a script, you must execute the script within the test for the variable values to be passed.

Deleting Local Variables From a Visual Test

Delete local variables when they are no longer needed in a visual test.

1. Open the visual test from which you want to delete local variables.
2. Perform one of the following steps:
   - To delete all local variables from the test, select the `<<Start>>` step and press `Delete`.
   - To delete a specific local variable from the test, select the `<<Start>>` step, select the variable in the **Properties** pane, and then click **Delete**.

Global Variables

Silk Test Workbench lets you create variables to represent data in a visual test so the data is available to any other visual test in the same database. You do this by setting the data to global variables.

Global variables are useful for data that needs to be used in several different tests, and the data to be used is constant. You can define the data in a global variable once, then reference the global variable in each visual test that requires the data. If the data represented by a global variable changes, you change the data in one place. Each visual test automatically uses the new data during its next playback.

To use global variables, you can create them in one visual test and store their values to local variables in other visual tests. This lets you create data in one test that can be re-used throughout your testing solution. Once data in a global variable is read into a local variable, the data can be used like any local variable.

Unlike local variables, global variables are not created and deleted only during playback of the visual test in which they are created. A global variable can be accessed and modified by any visual test that plays back the visual test containing the global variable.

Create a global variable in a visual test by creating a `Set global` step and setting the step's properties to define the variable. Each `Set global` step represents one global variable definition.

Managing Global Variables in Visual Tests

To use a global variable value in a visual test, the global variable must be stored to, and used as, a local variable defined in a script. Also, the visual test containing this variable must be inserted into any visual test that uses the value. This includes all steps of the visual test containing a global variable, which may produce unwanted test results.

Because of this, we recommended that all global variables for a test solution be created and maintained in one visual test. This has the following benefits:

- You can see each global variable definition used for a testing solution in one place.
- Duplicate global variable names can be eliminated.
- When the visual test containing the global variables is inserted into another visual test, only global variables are loaded to the calling script during playback. No unwanted additional steps are played back in the visual test containing the global variables.

If a local variable is used to store a global variable, any value previously set for the local variable is changed to the value of the global variable.

A global variable defined in a visual test can have the same name as a local variable. If you modify the global variable value, it does not modify the local variable value that has the same variable name. However, to avoid misuse of data in visual tests and to avoid confusion, ensure that global names are unique from local or other variable names.

Although two visual tests can create global variables of the same name, avoid this practice as well because of the possibility of using a global variable with an unexpected value.

Creating Global Variables For Visual Tests

Global variables are created for use within a visual test, but can be available to any visual test. A global variable is defined in a `Set global` step using the **Assignment** properties. Once a global variable is created, it is available to visual tests that execute the visual test in which the global variable resides.

1. Open the visual test in which you want to create a global variable.
2. Select the step that precedes where you want to create the global variable.
3. Choose **Insert** > **Variable** > **Set Global**. The `Set global variable` step is created after the selected step. Properties for the `Set global variable` step display in the **Properties** pane.
4. In the **Properties** pane for the `Set global variable` step, group properties by category.
5. Update the **Assignment** category properties to define the name of the global variable and any value to be assigned to it. Once the **Assignment** category properties are updated and the visual test is saved, the Set global variable step looks like the following: `Set global variable 'Global variable name' to 'value'`.
6. Repeat steps 2 through 5 to create other global variables as needed. Once global variables are set, they can be read into local variables in other visual tests. To do this, you must first insert the visual test (containing the global variable) into the visual test that contains the local variables.

Retrieving Global Variables For Use in a Visual Test

After global variables are created in a `Set global` step, they are available to any visual test that executes the visual test containing the `Set global` step. The global variable must first be read to a local variable created in the visual test that uses the global variable, and used as a local variable.

1. Open the visual test in which you want to use a global variable.
2. Create the local variable into which you want the global variable value to be stored.

   You can create local variables in any test step.
3. Create a step that executes the visual test where the global variable is created. Skip this step if the global variable is to be retrieved to a local variable that is defined in the same visual test.

   This allows the global variable to be accessed by the visual test in which you want to use the global variable.
4. Select the step that precedes the location where you want to retrieve the global variable.
5. Choose **Insert** > **Global** > **Get Global**. The `Get contents of global variable` step is created after the selected step. Properties for the `Get contents of global variable` step display in the **Properties** pane.
6. In the **Properties** pane for the `Get global` step, group properties by category.
7. Update the **Assignment** category properties to define the name of the global variable and any value to be assigned to it. After the properties are updated and the visual test is saved, the `Get contents of global variable` step appears as: `Get contents of global variable '[Global variable name]' and put into local variable '[Local variable name]'`
8. Repeat steps 2 through 7 to create other global variables. After a global variable is retrieved to a local variable, the local variable contains the global variable's value. Use it in visual test expressions and in the same manner as a local variable.

Reserved Variables

Silk Test Workbench exposes data about visual test playback that can be used during visual test playback. This data includes information about the success or failure of overall and specific playback results, current values input to controls, or text of the most recent error, or information about the last step executed. The exposed playback data can have many uses. For example, you can build conditions based on the success or failure of an executed step. The conditions can be used to determine how to react to the results during playback.

Visual test playback data is held in Reserved variables. Reserved variables are defined by Silk Test Workbench. They are read-only, which means their values can be returned and used in a visual test, but the values cannot be set by users. All reserved variables begin with a "*st_*" prefix.

Reserved variables can be returned as the value of any property that can use a variable as its value.

Reserved variables are bi-directionally inherited between visual tests executed in the same test. For example, if Visual test B is inserted in Visual test A, then Visual test B inherits the reserved variable values of Visual test A. Conversely, after Visual test B plays back, Visual test A inherits the reserved variable values of Visual test B.

Use the **Select a Variable** dialog box to select reserved variables for use in a visual test.

List of Reserved Variables

The following table lists reserved variables that return playback related data for a visual test:

Variable Name	Default Value	Type	Description
st_LastError	Empty string	Text	Returns the text of the last error generated during playback. If no errors are generated during playback (up to the point where the st_LastError value is returned) the value is `Nothing`.
st_LastReturnValue	False	Various	Return value of the last executed step in the visual test. The return value is based on the step type. Most automation steps return True if the operation is successful, and False if it is not. Automation or other steps that return a value such as a string, a column number, or a date, returns that information to the st_LastReturnValue variable.
st_LastValidation	False	Boolean	Returns the result of the last verification logic performed. True = most recent verification logic was successful. False = most recent verification logic failed.
st_Project	Project name where the executing visual test resides.	Text	Returns the name of the project where the executing visual test resides.
st_RepeatListValue	Empty string	Text	Returns the current input value of the most recently attached control, when an enumeration of that control has been performed. For example, when repeat logic is used to cycle through the items of a ListBox control, the variable can return the current item in the ListBox. This value returns the names of any items contained in a control. Use Repetition logic to cycle through the items in a list, and use st_RepeatListValue as the value to select.
st_VisualTest	Name of the executing visual test	Text	Returns the name of the executing visual test.

Reserved Variable Return Values

The following list the values returned for reserved variables and various step types:

Step Type	st_LastReturn Value	st_LastError	st_LastValidation	st_RepeatListValue
<<Start>>	True	Unchanged	Unchanged	Unchanged
<<End>>	True	n/a	n/a	n/a
Automation steps	The return value of the Open Agent method.	Contains any error message that the method sets.	Unchanged	Unchanged
Playback Visual Test	True = visual test is executed. False = visual test is not executed.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged
Verify	Returns the result of the last verification logic performed. True = successful verification of most recent. False = failed verification of most recent.	This may contain an error message if the evaluated expression fails, e.g., referencing a variable that no longer exists.	Unchanged	Unchanged
Get contents of global variable	True = retrieved variable contents. False = contents are not retrieved.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged
Set global variable	True = variable content is set. False = variable content is not set.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged
Get playback setting	True = playback setting is put into a local variable. False = playback setting failed to be put into a local variable.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged
Set playback setting...	True = playback setting is set. False = playback setting is not set.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged

Step Type	st_LastReturn Value	st_LastError	st_LastValidation	st_RepeatListValue
Run program	The return values of the ShellExecute API function (greater than 32 is success). This is similar to the VBA Shell() function.	st_LastReturnValue > 32 - Empty st_LastReturnValue <= 32 - Contains the error message returned from Windows, e.g., for 2 the message would be "The system could not find the file specified."	Unchanged	Unchanged
Message box	The number of the button that was pressed (Long): 1 = OK 2 = Cancel 3 = Abort 4 = Retry 5 = Ignore 6 = Yes 7 = No	This can contain an error message if the evaluated expression fails, e.g., referencing a variable that no longer exists.	Unchanged	Unchanged
Start timer	True = timer is being started. False = timer has already been started.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged
Resume timer	True = timer is being resumed after a Stop timer. False = timer is already running.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged
Reset timer	True	Empty	Unchanged	Unchanged
Stop timer	The elapsed time on the specified timer in milliseconds (Long). False (Boolean) if the timer was not started.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged
Delay	True	Empty	Unchanged	Unchanged
Go to label [label name]	True = step executed.	st_LastReturnValue = True - Empty.	Unchanged	Unchanged

Step Type	st_LastReturn Value	st_LastError	st_LastValidation	st_RepeatListValue
	False = step did not execute.	st_LastReturnValue = False - Contains an error message.		
On error	True = step executed.	st_LastReturnValue = True - Empty.	Unchanged	Unchanged
	False = step did not execute.	st_LastReturnValue = False - Contains an error message.		
Result Comment	True = step executed.	st_LastReturnValue = True - Empty.	Unchanged	Unchanged
	False = step did not execute.	st_LastReturnValue = False - Contains an error message.		
Using [locator name] (screen step)	True = step executed.	st_LastReturnValue = True - Empty.	Unchanged	Unchanged
	False = step did not execute.	st_LastReturnValue = False - Contains an error message.		
Corrupt item placeholder	False	Empty	Unchanged	Unchanged
Logic - If/ElseIf/ Else/EndIf	Unchanged	This may contain an error message if the evaluated expression fails, e.g., referencing a variable that no longer exists.	Unchanged	Unchanged
Loop - Repeat/ EndRepeat	Unchanged	This may contain an error message if the evaluated expression fails, e.g., referencing a variable that no longer exists.	Unchanged	Unchanged when a loop does not iterate the contents of list controls. When the loop iterates the contents of list type controls this contains both the index and value of the item as a compound value (represented as a formatting string in a visual test).
Stop	True	Empty	Unchanged	Unchanged
Label	Unchanged	Unchanged	Unchanged	Unchanged
Property from Control	True = found control and the variable set correctly.	st_LastReturnValue = True - Empty.	Unchanged	Is always set to the value of the property (even if no local variable is specified).

Step Type	st_LastReturn Value	st_LastError	st_LastValidation	st_RepeatListValue
	False = an error occurred, e.g., the control could not be found or the target variable was not found.	st_LastReturnValue = False - Contains an error message.		
Get contents of eCATT argument	True if the argument contents are retrieved, false if the contents are not retrieved.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged
Set eCATT argument	True if the argument content is set, false if it is not.	st_LastReturnValue = True - Empty. st_LastReturnValue = False - Contains an error message.	Unchanged	Unchanged

Sharing Data Between Visual Tests

Silk Test Workbench lets you pass data from one visual test to another. Create parameters to hold the data in one test. Then execute that test in another visual test to pass in the parameters and use their data.

Parameters are passed into any visual test that executes the test containing the parameters. Once passed in, you can assign data values and use parameters in expressions, test logic, or anywhere parameters can be used. You can also assign the parameter value to a local variable and use the local variable.

Create parameters to make data available in any visual test. In the visual test that uses the parameter, perform the following tasks:

- Insert the visual test that creates the parameter.
- Assign values to the parameters.

Adding Parameters to a Visual Test

Visual tests can receive data from a script or another visual test as an input parameter and can pass data to another visual test as an output parameter. Parameters are defined in the `<<Start>>` step of a visual test, edited in the **Properties** window, and are available to the visual test during playback. Once parameters are created, they are available to the visual test in which they are created.

1. Open the visual test for which you want to create parameters.
2. Click the **<<Start>>** step in the **Task** pane.
3. In the **Properties** pane, navigate to the **Variables** category.
4. Select the parameter type that you want to create:
   - To create a new input parameter, click the **Input parameters** field.
   - To create a new output parameter, click the **Output parameters** field.
5. Click **Add Parameter**. The **Add Parameter** dialog box opens.
6. Type a name for the parameter into the **Name** field.

   Valid characters for parameter names include uppercase and lowercase alphanumeric characters and the underscore ("ABC", "abc", "_"). If a parameter name being added or edited is a duplicate of another parameter name, supply a unique name for the parameter.

7. Select a data type for the parameter from the **Type** list.

   Silk Test Workbench supports the following parameter types:

Boolean	The parameter's value is either `True` or `False`.
Number (Double)	The parameter's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Number (Long)	The parameter's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Text	The parameter's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation.
Enumeration	This parameter groups together a set of values and orders them sequentially from 1 to *n*. Use this parameter type with properties or parameters that expect an enumeration data type. An enumeration stores the numeric value in a 64-bit integer in the range -9223372036854775808 to 9223372036854775807. However, the engine only supports 32-bit enumerations, while the UI supports 64-bit enumerations (in future releases, the engine will support 64-bit enumerations).

8. In the **Initial Value** field, type the value for the parameter to be passed.

   If left empty, the initial value of the parameter as set in the visual test is used. The value must be consistent with the type. For example, if the type is Number (Long), the initial value must be an integer.
9. Click **OK**. The name and initial value (if any) of the parameter display under the selected parameter type in the **Properties** pane for the `<<Start>>` step.
10. Repeat the preceding step to create other parameters as needed. The fields for the parameter types indicate the number of parameters of the type that are currently defined. For example, if the field for the `Number(Long)` item is **3 Items**, there are three parameters currently defined for the visual test that are of the `Long` data type.

Note: Parameters are not assigned a value when they are created. The parameters can only be assigned a value when the test is executed in another visual test. When playing back a visual test in which the parameters are created, values must be assigned when playing back the visual test that creates the parameters. In this scenario, the **Parameters** dialog box opens when playback starts, allowing values to be assigned to the parameters.

Assigning Values to Parameters in a Visual Test

To pass data between visual tests, one of the visual tests must be called as a test step from within the other visual test. We will refer to the calling visual test as the *host test* and to the called visual test as the *child test*. To pass parameters during execution from the child test to the host test, open the host test and assign values to the parameters in the child test.

Note: Before you can assign values to parameters in a visual test, the parameters need to be added to the visual test. For additional information, see *Adding Parameters to a Visual Test*.

Tip: To see the values of parameters passed between visual tests, add result comment steps into the visual test. For the comment **Expression** property, select **Variable**, then select a parameter from the list that contains the passed value.

1. Open the host test.
2. In the **Test Steps** pane, select the step that plays back the child test that contains the parameters.

The step text looks like `Playback visual test` [*asset name*], where [*asset name*] is the name of the child test.

Properties for the step display in the **Properties** pane. These properties include the parameters that have been set up as visual test parameters. These are in the categories **Input parameters** and **Output parameters**. Each parameter is listed by its data type. The property name for each parameter is its parameter name.

If the parameters do not display in the **Properties** pane, reload the child test. To do this, select the **Name** property and click **Refresh Parameters** to update the list of parameters.

3. Select a parameter.
4. Click **Edit Parameter**. The **Edit Parameter** dialog box opens.
5. Type a value for the parameter into the **Initial Value** field,
6. Repeat the previous step to assign values to other parameters.

Editing a Visual Test Parameter

1. Open the visual test for which you want to create parameters.
2. Click the **<<Start>>** step in the **Task** pane.
3. In the **Properties** pane, navigate to the **Variables** category.
4. Navigate to the parameter type that you want to edit:
   - If you want to edit an input parameter, navigate to **Input parameters**.
   - If you want to edit an output parameter, navigate to **Output parameters**.
5. Select the parameter that you want to edit.
6. Click **Edit Parameter**. The **Edit Parameter** dialog box opens.
7. Type a name for the parameter into the **Name** field.

   Valid characters for parameter names include uppercase and lowercase alphanumeric characters and the underscore ("ABC", "abc", "_"). If a parameter name being added or edited is a duplicate of another parameter name, supply a unique name for the parameter.
8. Select a data type for the parameter from the **Type** list.

   Silk Test Workbench supports the following parameter types:

Boolean	The parameter's value is either `True` or `False`.
Number (Double)	The parameter's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Number (Long)	The parameter's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Text	The parameter's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation.
Enumeration	This parameter groups together a set of values and orders them sequentially from 1 to *n*. Use this parameter type with properties or parameters that expect an enumeration data type. An enumeration stores the numeric value in a 64-bit integer in the range -9223372036854775808 to 9223372036854775807. However, the engine only supports 32-bit enumerations, while the UI supports 64-bit enumerations (in future releases, the engine will support 64-bit enumerations).

9. In the **Initial Value** field, type the value for the parameter to be passed.

   If left empty, the initial value of the parameter as set in the visual test is used. The value must be consistent with the type. For example, if the type is Number (Long), the initial value must be an integer.
10. Click **OK**.

Deleting Parameters from a Visual Test

1. Open the visual test that contains the parameters that you want to delete.
2. Click the **<<Start>>** step in the **Task** pane.
3. In the **Properties** pane, navigate to the **Variables** category.
4. Click the text box for the parameter that you want to delete, and then click **Delete**.

Consuming Parameters from Silk Central

To enable Silk Test Workbench to use a parameter that has been set for a test in Silk Central, define a variable in Silk Test Workbench that has the same name as the parameter in Silk Central and then use the variable.

Passing Data Between Visual Tests

Silk Test Workbench enables you to pass data between visual tests. This lets you leverage data created in your existing Silk Test Workbenchvisual test base, as well as allowing you to use the most efficient testing assets and still retain data integrity within a testing solution. Pass data between visual tests using visual test parameters.

To pass data between visual tests, one of the visual tests must be called as a test step from within the other visual test. We will refer to the calling visual test as the *host test* and to the called visual test as the *child test*. To pass data between visual tests, the visual tests must be in the same project, or the child test must be in a referenced project.

With visual test parameters, you can:

- Create data in a visual test and use it in another visual test.
- Create data in a visual test, pass it to a child test, modify the data in the child test, and pass it back to the host test.

The process is as follows:

1. Add input and output parameters to the child test.
2. Insert the child test as a test step into the host test.
3. Pass data from the child test to the host test by editing the value of the input parameter in the host test.

   Note: The value that you send from the child test is not saved in the host test, but it can be used in computations in the host test.
4. In the host test, create a local variable to store the data passed from the output parameter of the child test.
5. Use the local variable within the host test.

Note: For a detailed example of how to create input and output parameters and how to use these paremeters in a visual test, refer to *Playing Back Scripts From Visual Tests* in the *Welcome to the SilkTest Workbench Visual Test Tutorial*.

Defining Parameters to Pass Between Visual Tests

When calling a visual test as a test step from another visual test, you can pass parameters between these visual steps. We will refer to the calling visual test as the *host test* and to the called visual test as the *child test*.

- To pass a parameter from the host test to the child test, you have to specify the parameter as an input parameter for the child test.
- To pass a parameter from the child test to the host test, you have to specify the parameter as an output parameter for the child test.

1. If you want to pass parameters from the host test to the child test, add the parameters as input parameters to the child test:
   a) Open the child test.
   b) Select the **Start** step.
   c) In the **Properties** pane, select **Input Parameters**.
   d) Click **Add Parameter**. The **Add Parameter** dialog box opens.
   e) In the **Parameter Name** field, type a name for the new parameter.
   f) From the **Type** list, select the data type that you want to pass to the visual test.
   g) Leave the default value in the **Initial Value** field.
   h) Click **OK**. The input parameter displays in the list of input parameters in the **Properties** pane.

   Note: Input parameters defined for a visual test do not appear in the visual test steps. You can view the input parameters in the **Properties** pane of the **Start** step.

   Note: The value that you send from the host test is not saved in the child test, but you can use the value in computations in the child test.
2. If you want to pass parameters from the child test to the host test, add the parameters as output parameters to the child test:
   a) Open the child test.
   b) Select the **Start** step.
   c) In the **Properties** pane, select **Output Parameters**.
   d) Click **Add Parameter**. The **Add Parameter** dialog box opens.
   e) In the **Parameter Name** field, type a name for the new parameter.
   f) From the **Type** list, select the data type that you want to pass from the visual test.
   g) Leave the default value in the **Initial Value** field.
   h) Click **OK**. The output parameter displays in the list of output parameters in the **Properties** pane.

   Note: Output parameters defined for a visual test do not appear in the visual test unless they are actually declared in the visual test. You can view the output parameters in the **Properties** pane of the **Start** step.
   i) Open the host test.
   j) Add a local variable for any parameter that you want to pass from the child test to the host test.
3. To view up-to-date parameters of the child test, click the **Name** text box in the **Properties** pane of the **Start** step of the child test, and then click **Refresh Parameters**.

Using Parameters Passed From a Visual Test in Another Visual Test

You can pass the output parameters of a visual test to other visual tests that call this visual test. We will refer to the calling visual test as the *host test* and to the called visual test as the *child test*. To use the parameters of a child test in a host test, assign the output parameters of the child test to local variables in the host test.

1. Open the host test.
2. Create a local variable for each output parameter of the child test that you want to pass to the host test.
   For additional information, see *Adding Local Variables to a Visual Test*.
3. In the **Test Steps** pane, select the step that executes the child test. The step text looks similar to `Playback visual test 'asset name'`, where 'asset name' is the name of the child test.
4. In the **Properties** pane for the step, expand the **Output parameters** category.
   The names of the output parameters of the child test display in the **Output parameters** subcategories.

If the parameters do not display in the **Properties** pane, click the **Name** text box, and then click **Update Parameter Names**.

5. Click the text box of an output parameter name and select the appropriate local variable from the list.
6. Repeat the preceding step for each output parameter of the child test that you want to pass to the host test.
7. *Optional:* To see the values of the output parameters that are passed from the child test to the host test, insert a result comment step for each parameter into the host test.
   a) In the host test, right-click on the step after which you want to insert a result comment.
      For example, click on the step that calls the child test.
   b) Select **Insert** > **Result Comment**.
   c) In the properties of the new step, click into the **Expression** text field.
   d) Click **Select**.
   e) Select **Variable** from the list. The **Select a variable** dialog box opens.
   f) Select the variable that corresponds to the output parameter of the child test.
8. Modify the appropriate test steps in the host test to include the local variable.
   a) Select the step in the **Test Steps** pane.
   b) In the **Properties** pane, expand the **Parameters** section.
   c) Click into the text box that corresponds with the property type of the local variable.
      For example, if the local variable is of the type *text*, and you want to change a test step that types text into a text field in the AUT, click into the **keys** text box, which also is of the type *text*.
   d) Click **Select**, and then select **Variable**. The **Select a Variable** dialog box opens.
   e) Select the local variable that you want to use, and then click **OK**.

If the child test has been included for playback in the host test and its input and output parameters have been defined correctly, the child test can now be played back and parameters are successfully passed between the child test and the host test.

Note: For a detailed example of how to create input and output parameters and use them in a visual test, refer to *Playing Back Scripts From Visual Tests* in the *Welcome to the SilkTest Workbench Visual Test Tutorial*.

Managing Test Flow

Typically during visual test playback, Silk Test Workbench executes steps in sequential order. However, in some situations executing steps in a non-sequential order may be useful. For example, a series of steps may need to be executed at several different points in a visual test before the visual test completes. Rather than repeating the series of steps in several different places in the test, each point can navigate to the series of steps placed at the end of the test.

Navigating to a specific step is also a useful means of exiting repetition logic that repeats when a certain condition within the repetition is met.

Use labels in visual tests to "bookmark" specific areas of a visual test to aid in test navigation. You can create test steps to go to any label defined in a visual test. Since labels are simple marker steps in a visual test, they can also provide a way to describe a subsequent series of steps that is viewable from the **Test Steps** pane.

Executing a Script Within a Visual Test

Visual tests can call and execute existing .NET scripts. This allows a modularized approach to application testing and provides greater control over test execution. When executing a script within a visual test, it is important to remember to keep test site integrity and ensure any test applications being tested are in the correct initial playback state.

You can also use this procedure to create a driver test.

1. Open the visual test in which you want to insert the script.
2. Select the test step that precedes where you want the inserted script to execute.
3. Choose **Insert** > **.NET Script**. The **Browse for .NET Script** dialog box opens.
4. Select the script to insert and then click **OK**. Silk Test Workbench inserts a step below the selected step. The inserted step calls and plays back the selected script. The step text is as follows:

   ```
   Playback .NET Script 'asset name'
   ```

   where 'asset name' is the name of the inserted script.

During playback, when the preceding step executes, the inserted script plays back to completion before the next step in the visual test executes. After a script has been inserted into a visual test, you can edit it and configure the settings used to play it back.

The tested applications are configured, hooked, and started when required, and remain hooked until the end of the playback session. For example, if the visual test has an application configuration for Notepad, and the .NET script has an application configuration for Calculator, the following happens:

- Notepad is hooked and started when the execution of the visual test starts.
- Calculator is hooked and started when the execution of the inserted .NET script starts. Notepad remains hooked.
- Both Notepad and Calculator remain hooked when the execution of the inserted .NET script stops.
- Both Notepad and Calculator get unhooked when the playback session ends, which means that the execution of the visual test stops.

Modifying a Script that is Inserted in a Visual Test

After a script has been inserted into a visual test, you can edit it and configure the settings used to play it back.

1. Open the script that you want to change, modify it, and then save the script.
   For example, you might modify the input or output parameters in the script.
2. Open the visual test in which you inserted the script that you modified.
3. In the **Test Steps** pane, select the step in the visual test that executes the script. The step text looks similar to `Playback .NET Script 'asset name'`, where *asset name* is the name of the inserted script.

   Properties for the step display in the **Properties** pane. This includes any input and output parameters for the script.
4. In the **Properties** pane for the step, expand the **.NET Script** category.

   The name of the inserted script displays in the **Name** row.
   a) To change the script that is inserted in the visual test, click **Browse for .NET Scripts** and select the script that you want to use.
   b) To update the parameter names only, click **Update Parameter Names** and then click **Yes** when prompted.
   c) To reload the available parameters, click **Refresh Parameters** and then click **Yes** when prompted.
   d) To change the playback settings, select another value from the **Playback settings** list.

Creating a Message Box in a Visual Test

Visual tests can display messages to a tester during playback. For example, use message boxes to manually verify test outputs, view information about controls, or communicate any other information during testing.

To create a message box, insert a message box step into a visual test. During playback, the message box displays with the message content you specified when creating the step.

1. Open the visual test in which you want to insert the message.
2. Select the step that precedes where you want the message box to display.
3. Choose **Insert** > **Message Box**. Silk Test Workbench inserts a `Message box` step below the selected step.
4. In the **Properties** pane for the `Message box` step, click the icon to group properties by category or to sort alphabetically.
5. Edit the **General** category properties in the **Properties** pane to specify the message box title and text.
6. Edit the **Advanced** category properties in the **Properties** pane to specify display elements (such as buttons and icons) of the message box.

Creating a Label Test Step for Visual Test Navigation

Label test steps do not execute a test action. Use label test steps for program flow management as the target for "go to" steps, and to describe a subsequent series of steps.

1. Open the visual test in which you want to insert the label.
2. Select the step that precedes where you want the label step.
3. In the **Test Steps** pane, click **Actions** > **Insert** > **Label**. The label step is created after the selected step. The **Properties** window for the step displays the **Label** property for the label step.
4. Click the value area of the **Label** property for the label step and type descriptive text for the label.

   Use the descriptive text to describe the label step's use. If using the label as the destination of a "go to" step, consider descriptive text that explains why program flow branches to this step. If using the label to describe a subsequent series of steps, use the text to describe the purpose of the subsequent series of steps.

   You can change the **Label** property value. However, any steps that reference the Label step, such as logic or "go to" steps must be manually updated to reference the Label step by its new value.
5. Click **Save** to save the label step and its **Label** property value. The updated step and its descriptive text appear in the visual test.
6. If you use the Label step for navigation, create "go to" steps as needed that navigate to the label step.

Creating a Go To Test Step for Visual Test Navigation

Go to test steps do not execute a test action. Use go to test steps for program flow management to jump to a specified label step during test playback.

1. Open the visual test in which you want to insert the go to step.
2. Select the step that precedes where you want to insert the go to step.
3. Choose **Insert** > **Program Flow** > **Go To**. The go to step is created after the selected step. The **Properties** window for the step displays the **Label** property for the Label step.
4. Click the value area of the **Label** property for the go to step and select the destination label from the list.

   Tip: To use a label step as a destination for the go to step, first create the label step.
5. Click **Save** to save the go to step and its **Label** property value.

   The updated step and its descriptive text display in the visual test.

Starting an External Application Within a Visual Test

Visual tests can start external applications during playback. This enables you to test the startup of test applications to ensure they start as expected, while also ensuring the state of a test site when testing against applications when they are started.

1. Open the visual test in which you want to start the external application.

2. Select the step that precedes where you want the application to start.
3. Choose **Insert** > **External Program**. Silk Test Workbench inserts a `Run program` step below the selected step.
4. Select the newly created `Run program` step to display its properties in the **Properties** pane. The **Command** category properties display in the **Properties** pane.
5. In the **Path Name** text box, perform one of the following steps:
   - Type the path to the external program that you want to start.
   - Click **Select file** to open the Select file dialog box and navigate to the external program that you want to start.
   - Click **Select** and specify a literal value, variable, expression, or ActiveData to use to start the external program.

   Note: If the operation to perform is **Find** or **Explore**, only the a path name is required for this property.
6. From the **Operation** list box, select the action to perform when the application starts.

   You can choose one of the following:

   - **Open** (default) – Opens the specified file using any named parameters. The file can be an application, a document file, or a folder.
   - **Edit** – Starts an editor and opens the file for editing. If the file specified by the **Path Name** property is not a document file that can be opened in a document editor, the operation does not execute.
   - **Explore** – Uses Windows Explorer to open the folder specified by the **Path Name** property.
   - **Find** – Starts Windows Explorer in search mode starting from the specified directory. The directory can be specified using either the **Path Name** property or the **Directory** property. If using the**Path Name** property and including a file name, playing back the test does not execute the step.
   - **Print** – Prints a document file specified by the **Path Name** property. If the file specified by the **Path Name** property is not a document file, playing back the test does not execute the step.
7. In the **Parameters** text box, specify any parameters to pass to the application for startup.

   If the file specified by the **Path Name** property is an application file, use this property to specify a null-terminated string containing parameters passed to the application and used for startup. If the file specified by the **Path Name** property is a document or other type of file, leave the value blank.
8. In the **Directory** text box, specify a default directory that can be used for the action specified by the **Operation** property.
9. From the **Window style** list box, select how the application or program displays when it starts.

   Values include:

   - **Hide** – Window is hidden when program or application starts and focus is passed to another window.
   - **Maximize** – Maximizes the application window.
   - **Minimize** – Minimizes the application window at startup and activates the next top-level window in the z-order.
   - **Restore** – Starts and displays the application window. If the window is minimized or maximized, then it is restored to its original size and position. Use this value when restoring a minimized window.
   - **Show** – Starts the application window and displays it in its current size and position.
   - **Show default** – Sets the window display state based on the information provided by the program that starts the application.
   - **Show maximized** – Starts the application and displays it in a maximized window.
   - **Show minimized** – Starts the application and displays it in a minimized window.
   - **Show minimized, do not activate** – Displays the application as a minimized window. The current active window remains active.
   - **Show in current state, do not activate** – Displays the application in its current window state. The active window remains active.
   - **Show in most recent position, do not activate** – Displays the application in its most recent window size and position. The active window remains active.

- **Show normal** – Starts the application and displays a window. If the window's most recent position minimized or maximized, the window is restored to its original size and position. Use this value when displaying the window for the first time.

Inserting a Wait For Object Step in a Visual Test

You can insert a step anywhere in a visual test to wait for an object to exist or disappear during visual test playback. Include a wait for object step to coordinate synchronization between the visual test playback and the test application.

1. Open the visual test in which you want to include a wait for object step.
2. Perform one of the following steps.
   - Select the step that precedes where you want the wait for object step to execute and then choose **Insert** > **Synchronization and Timing** > **Wait for object**.
   - Open the **Logic Toolbox**, drag **Wait for objects to appear or disappear during playback of this visual test** to the appropriate position in the **Test Steps** pane.

   Silk Test Workbench inserts a `Wait for " to exist` step below the selected step.
3. Select the newly created `Wait for " to exist` step to display its properties in the **Wait** category in the **Properties** pane.
4. In the **Properties** pane, in the **Locator** text box, identify the object that you want to wait for.
   - **Application Under Test** – Click this button to identify a visible control directly from the application under test.
   - **Screen Preview** – Click this button to identify a control directly from the **Screen Preview**, if the application under test is not available.
   - **Identify Object Dialog** – Click this button to use the **Identify Object** dialog box to identify a non-visible control. The **Identify Object** dialog box additionally enables you to edit the locator of the control.
   - **Select** – Click this button to assign a literal, variable, expression result, or ActiveData value.
5. From the **Wait type** list, specify whether to wait for the object to exist or disappear.
6. In the **Timeout** text box, type the maximum number of milliseconds to wait for the object to exist or disappear.

   By default, this value is set to the **Default wait timeout (milliseconds)** option value in the **Options** dialog box. If no object matches within the timeout period, a playback error occurs.
7. Click **Save**.

Waiting for a Property

Insert a step into a visual test to wait for a property of a control in the application under test (AUT) to get a specific value. Insert such a step to coordinate synchronization between the visual test playback and the test application.

1. Open the visual test to which you want to add the step.
2. Right-click on the step that precedes where you want to insert the step and choose **Insert** > **Synchronization and Timing** > **Wait for property**. Silk Test Workbench inserts a new step below the selected step.
3. Select the newly created step to display its properties in the **Properties** pane.
4. In the **Locator** field, specify the locator of the control in the AUT which contains the property that you want to wait for.
5. In the **Property** field, select the property.
6. In the **Value** field, specify the value for which you want to wait.
7. In the **Timeout** field, specify the maximum time to wait before continuing playback.

8. Click **Save** to save the step and its property value. The updated step and its descriptive text appear in the visual test.

Setting a Playback Delay for a Visual Test

You can insert a delay step anywhere in a visual test to delay the visual test playback. You control the amount of time for the delay. You can insert any number of delays into a visual test. Use delays for coordinating synchronization between the visual test playback and the test application.

1. Open the visual test in which you set a playback delay.
2. Select the step that precedes where you want the delay step to execute.
3. Choose **Insert** > **Synchronization and Timing** > **Delay**. Silk Test Workbench inserts a `Delay for 1 second(s)` step below the selected step.
4. To change the delay increment and the time amount, select the newly created `Delay for 1 second(s)` step to display its properties in the **Properties** pane.
   a) From the **Delay type** list, select **Seconds** or **Milliseconds** for the delay interval.
   b) In the **Delay amount** text box, specify the amount of time for the delay.
      Type a literal value or click **Select** to assign a variable, expression result, or ActiveData value.
5. Click **Save**.

Inserting Timers in a Visual Test

You can insert a timing step anywhere in a visual test to start, stop, or resume a timer during playback. Including timers enables you to determine how much time specific steps take to playback.

1. Open the visual test in which you want to include a start, stop, or resume timer.
2. Select the step that precedes where you want to insert the timer and then choose **Insert** > **Synchronization and Timing** and then choose one of the following commands:
   - **Start Timer** – Choose this option to specify the first step in the sequence of steps for which you want to time playback.
   - **Stop Timer** – Choose this option to specify the last step in the sequence of steps for which you want to time playback.
   - **Resume Timer** – Choose this option to resume timing a sequence of steps. For instance, if you do not need to record the time for several test steps, you can insert a stop timer and then insert a resume timer for the steps that you want to include in the timing.

   Silk Test Workbench inserts a `Start timer number n`, `Stop timer number n`, or `Resume timer number n` step below the selected step.
3. Select the newly created step to display its properties in **Timer** category in the **Properties** pane.
4. From the **Timer action** list, select the type of timer to use.
5. In the **Timer number** text box, type the number to assign to the timer or select one from the list.
   When timing multiple conditions, you can differentiate the timers by assigning a unique number to each timer. You can include up to 10 timers in a visual test.
6. Click **Save** to save the step and its property value. The updated step and its descriptive text appear in the visual test.

When you playback the visual test, the **Details** tab of the **Results** window shows the timer results in the **Result Detail** column of the test step.

Test Steps

Steps (also known as *test steps*) are the core executable units in a visual test. Steps are similar to code lines in scripts.

Test steps can only be executed using the **Test Steps** pane in the **Visual Navigator**.

Step flow and logic in a visual test is similar to code line flow and logic in a script. For example, a visual test must attach to controls in the test application before actions can be executed against them. Like scripts, visual tests contain automation objects that represent controls in the test application. The automation objects expose properties and methods that can be executed against the controls. Each automation action is represented as a step in a visual test.

Selecting Multiple Steps

You can update information for any group of steps in a visual test by selecting them as a group.

1. To select a group of sequential test steps, perform the following steps:
   a) Select the first test step in the sequential group of steps.
   b) Press the `Shift` key and select the last step in the sequential group. Or, choose**Edit** > **Select All** to select all steps in a visual test.
2. To select a group of non-sequential test steps, perform the following steps:
   a) Select the first test step in the group of steps.
   b) Press the `Ctrl` key and select random steps. Steps do not have to be in a continuous sequence.

Copying and Pasting Test Steps

You can copy and paste any test steps other than the <<`Start`>> and <<`End`>> steps into an open visual test. When copying and pasting test steps, consider the following:

- Silk Test Workbench copies test steps to the clipboard in the sequential order they appear in a visual test and not the order in which you select them. For example, if you select test step 5 before selecting test step 2, and then copy both to the clipboard, when you paste these test steps into a visual test, Silk Test Workbench pastes test step 2 followed by test step 5.
- Copying a Screen test step does not copy the associated test steps.
- To ensure the uniqueness of Label test steps, Silk Test Workbench appends "_1" when copying and pasting a Label test step.
- Copying a Repeat or End Repeat test step also copies the associated End Repeat or Repeat test step.
- Copying an `If` or `End If` test step copies any associated decision logic test steps (`If, Else, Else If, End If`), however, the dependent test steps contained in the decision logic test steps are not copied.
- You cannot copy multiple `Else` or `Else If` test steps to the clipboard unless you copy all decision logic test steps (`If, Else, Else If, End If`). When pasting an `Else If` test step, Silk Test Workbench only allows you to paste the step in the proper context.

1. With the visual test open in the **Test Steps** pane, select the step or steps to copy.

   You can select multiple sequential or non-sequential steps, and then copy and paste them as a sequential group.
2. Choose **Edit** > **Copy**.

   Paste the copied step(s) into a different visual test by opening that visual test in which you want to paste the copied steps. To move the selected step(s) instead of copying them, choose **Edit** > **Cut**.
3. Select the test step that you want the copied steps to appear after.
4. Choose **Edit** > **Paste**. Silk Test Workbench pastes the copied test steps into the visual test after the selected step.

Deleting Test Steps Using the Screen Preview

Since captured test application images in the **Screen Preview** are associated with test steps, when deleting any captured image using the **Screen Preview**, you can also delete any associated test steps from the **Test Steps** pane. This includes:

- All associated automation test steps.
- All other types of single-node test steps.
- All verification test steps.

You cannot delete decision and repetition logic test steps associated with the captured image being deleted from the **Screen Preview**. This ensures the logic is preserved and the visual test plays back correctly. You must manually delete decision and repetition test steps directly from the **Test Steps** pane.

1. In the **Test Steps** pane, select the test step of the captured image to delete. In the **Screen Preview**, the captured image appears.
2. Select the captured image in the **Screen Preview**.
3. Press `Delete`. A message appears asking you to delete the screen item and its associated steps.
4. Perform one of the following steps:
    - Click **Yes** to delete the captured image and its associated test steps in the **Test Steps** pane.
    - Click **No** to delete just the captured image in the **Screen Preview** and keep the associated test steps.
    - Click **Cancel** to close the message without deleting any thumbnails.

Deleting Test Steps Using the Test Steps Window

You can delete any test steps other than the `<<Start>>` and `<<End>>` steps from the **Test Steps** pane, however, since some test steps depend on other test steps for proper execution, the following rules apply:

- Deleting an `If` logic type test step also results in the deletion of any `Else If`, `Else`, or `End If` test step associated with the deleted `If` test step.
- Deleting an `End If` logic type test step also results in the deletion of any `If`, `Else`, or `Else If` test step associated with the deleted `If` test step.
- Deleting an `Else` or `Else If` logic type step only deletes the `Else` or `Else If` test step. The `If` and `End If` test steps are not deleted.
- Deleting a `Repeat` logic type test step also results in the deletion of the `End Repeat` test step.
- Deleting a test step that attaches to a screen or is an automation step results in the deletion of any associated steps with the exception of decision and repeat logic test steps. Automation steps associated with the attach step are deleted, unless another similar attach step to the same screen exists prior to the associated action steps.

1. With the visual test open in the **Test Steps** pane, right-click the test step or multiple test steps that you want to delete and choose **Delete**.

    You can select multiple sequential or non-sequential steps, and then delete them as a sequential group.
2. If any of the test steps are screen type steps, a message appears asking you to delete the screen item and its associated steps.
    - Click **Yes** to delete the selected step and its associated steps.
    - Click **No** to delete only the selected step and not delete any associated test steps.
    - Click **Cancel** to close the message without deleting any test steps.

Deleting Test Steps Using the Storyboard

Since thumbnails are associated with a group of test steps, you can also delete any associated test steps from the **Test Steps** pane when deleting a thumbnail using the **Storyboard**. This includes:

- All associated automation test steps.
- All other types of single-node test steps.
- All verification and event logic test steps.

You cannot delete decision and repetition logic test steps associated with the captured image being deleted from the **Screen Preview**. This ensures the logic is preserved and the visual test plays back correctly. You must manually delete decision and repetition test steps directly from the **Test Steps** pane.

1. In the **Storyboard**, click the thumbnail that corresponds to the test step or steps to delete.
2. Press `Delete`. A message appears asking you to delete the screen item and its associated steps.
3. Perform one of the following steps:
   - Click **Yes** to delete the thumbnail and its associated test steps in the **Test Steps** pane.
   - Click **No** to delete just the thumbnail in the **Storyboard** and the keep the associated test steps.
   - Click **Cancel** to close the message without deleting any thumbnails.

Enhancing Scripts

This section describes how you can modify VB .NET scripts to perform more advanced functions.

Referencing a Script from within a Script

You can create a script of often-used functions and re-use that script from any other scripts. You do this using the **Add .NET Script Reference** command.

Adding a script reference

1. Create a single script that you want to re-use or reference from another script.
2. Create a new driving script that will call your re-used script.
3. Expand the **Properties** tab.
4. Right click and select **Add .NET Script Reference**.
5. Select the script that you want to use click **OK**.

Script example

The following sample contains a script file called *Calculator* that will be referenced from another script.

Note: Since the following has no `Main()` method, it cannot run on its own.

```
Public Class Calculator
  Public Shared Function Add(left As Integer, right As Integer)
          Return left + right
     End Function

  Public Shared Function Subtract(left As Integer, right As Integer)
          Return left - right
     End Function

  Public Shared Function Multiply(left As Integer, right As Integer)
    Return left * right
  End Function

  Public Shared Function Divide(left As Integer, right As Integer)
    Return left / right
  End Function
End Class
```

Create another script, and add *Calculator* as a reference (expand the **Properties** tab, right click, select **Add .NET Script Reference**, and select **Calculator**) , and use the following code:

```
Public Module Main
    Dim _desktop As Desktop = Agent.Desktop
```

```
    Public Sub Main()
        MsgBox("2 + 3 = " + Calculator.Add(2, 3).ToString())
    End Sub
End Module
```

Adding a Verification to a Script while Recording

Do the following to add a verification to a script during recording:

1. Begin recording.
2. Move the mouse cursor over the object that you want to verify and press **Ctrl+Alt**.

   When you are recording a mobile Web application, you can also click on the object and click **Add Verification**.

   This option temporarily suspends recording and displays the **Test Logic Designer** wizard.
3. Select the object that you want to verify after the wizard opens. If you select a field with compound properties (**Position**, for example), code similar to the following will be added to your script: `Workbench.Verify(New TextPosition(1, 0), .TextField("TextField").Position)`.
4. Follow the wizard through the process and click **Finish** to close the wizard and continue recording.

Adding a Verification to a Script with Code

Insert a verification to determine if the script result matches the expected result.

1. Open the script in which you want to include a verification.
2. Insert one of the following verification methods:

   - To compare the expected value with the actual value, type:

     ```
     Workbench.Verify(expected As Object, actual As Object)
     ```

     For example, `Workbench.Verify("red", "red")` passes. While `Workbench.Verify("red", "green")` fails with the message `Actual: [green]; Expected: [red]`.
   - To compare the expected value with the actual value and add a comment, type:

     ```
     Workbench.Verify(expected As Object, actual As Object, comment As String)
     ```

     For example, `Workbench.Verify("red", "green", "checking colors")` fails with the message `checking colors - Actual: [green]; Expected: [red]`.
   - To compare the expected value with the actual value, to add a comment, and to add a screenshot to the result file if the verification fails, type:

     ```
     Workbench.Verify(expected As Object, actual As Object, comment As String,
     verifyFlags As VerifyFlags)
     ```

     For example, `Workbench.Verify("red", "green", "checking colors", verifyFlags.ScreenShotOnFailure)` fails with the message `checking colors - Actual: [green]; Expected: [red]` and adds a screenshot to the result file.
   - To verify the value returned by the expected result, type:

     ```
     Workbench.Verify(condition As Boolean)
     ```

     For example, `Workbench.Verify(True)` passes. While `Workbench.Verify(False)` fails.
   - To verify the value returned by the expected result and add a comment, type:

     ```
     Workbench.Verify(condition As Boolean, comment As String)
     ```

     For example, `Workbench.Verify(True, "Test Passed")` passes. While `Workbench.Verify(False, "Test Failed")` fails.

- To verify the value returned by the expected result, to add a comment, and to add a screenshot to the result file if the verification fails, type:

  ```
  Workbench.Verify(condition As Boolean, comment As String,
  verifyFlags As VerifyFlags)
  ```

  For example, `Workbench.Verify(True, "Test Passed", verifyFlags.ScreenShotOnFailure)` passes and adds no screenshot. While `Workbench.Verify(False, "Test Failed", ScreenShotOnFailure)` fails and adds a screenshot to the result file.
- To compare the actual value with the expected value for IEnumerable objects, such as lists and arrays, type:

  ```
  Workbench.Verify(expectedEnumerable, actualEnumerable)
  ```

  For example:

  ```
  Dim selectedItemsList = listBox.SelectedItems ' we assume that a list
  with the items "red" and "blue" is returned
  Dim expectedItemsList = New List(Of String)()
  expectedItemsList.Add("red")
  expectedItemsList.Add("blue")

  Workbench.Verify(selectedItemsList, expectedItemsList)  ' verification
  passes

  Dim expectedItemsArray = New String() { "red", "blue" }
  Workbench.Verify(selectedItemsList, expectedItemsArray) ' verification
  passes
  ```

  Note: Two IEnumerable objects are considered equal if both have the same number of elements, and the elements are equal and in the same order.

  Note: Mathematical operations with floating point numbers may lead to two numbers that are not completely identical because of their internal representation, although they should be equal from a user point of view. Therefore, floating point numbers (Double, Single) are considered equal if their difference is less than 0.00001. At times, this value may not be correct for the situation. In this case, compare the two values with the `Verify(result As Boolean)` instead.

Reusing Test Data in Scripts

Lists the types of reuse that are available to scripts.

Variables in Scripts

Since Silk Test Workbench uses VB.NET as its scripting language, you can use VB.NET variables in scripts. A *variable* is a named alias that contains data that can be used and modified during playback.

During recording, any data input against a control is recorded as literal data. For example, when a WPF TextBox control has a value typed into it, the resulting code might look like the following:

```
.WPFTextBox("@automationId='CarType'").SetText("Ford")
```

This works, but the script can never use data other than the value *Ford* in the TextBox control. However, if *Ford* is replaced with a variable that represents that data, then whatever data is used in the variable will be placed in the TextBox control.

```
.WPFTextBox("@automationId='CarType'").SetText(sCarmake)
```

Using variables provides greater testing flexibility because data used in scripts does not have to be constant.

If you declare a variable without a type, Silk Test Workbench generates a message indicating that no type was specified. Silk Test Workbench then uses the `Object` data type is then used by default. You can use any acceptable VB.NET data type as a variable data type. Rules regarding declaration of VB.NET variables apply when using them in scripts.

For example:

```
Dim dSalesPrice As Double

Dim iQuantity As Integer

Dim sCarmake As String
```

When Silk Test Workbench records data input into controls, if the value recorded is in quotes (""), it is usually a `String` data type and can be replaced with a variable of the `String` data type.

Public and Private Variables in Scripts

Functions and variables can be declared in scripts as `Public` and `Private`. A variable declared `Private` is visible only to the asset it is created in. So if a variable is created in a script or function, then it can only be used inside that asset, and not any others that may be called by the asset.

A variable declared `Public` is available to all assets after it is created. Any data passing from one function to another or from one script to another should be declared as a `Public` variable. However, use of public variables can create a situation where a publicly declared variable may be used in other scripts that may playback at the same time. This might cause corrupted data use or compile errors.

Variables should only be declared as `Public` when there is certainty that the same variable name not being used in other assets.

Public and private variables in scripts are used in the same way global and local variables are used in visual tests.

Declaring Variables in Scripts

Variables may be declared using the `Dim`, `Private`, or `Public` statements. The syntax is:

```
Dim VariableName As [data type]

'or

Private VariableName As [data type]

'or

Public VariableName As [data type]
```

For example, to declare a variable of type String and assign a value to that variable, you might type:

```
Dim carmake As String
carmake = "Honda"
```

Alternatively, the following code creates a variable of the type String and assigns a value to that variable. The variable has the type String because the compiler knows that "Honda" must be a String. This code achieves the same result as the preceding example and in most cases, the following example is preferable:

```
Dim carmake = "Honda"
```

If you declare a variable without a type in a script, Silk Test Workbench displays a message stating that no type was specified for the variable. The compiler assumes that the variable is of type `Object` by default.

It is recommended to declare all variables in scripts. Not declaring variables can cause errors. For example, a spelling mistake can cause two variables to exist instead of one, causing your script to malfunction. This is a very common error but often difficult to diagnose.

Note: You do not have to manually force variable declaration. Variable declaration is forced by default. You cannot use the `Option Explicit` statement to manually disable the enforcement of variable declaration in a script, because Silk Test Workbench performs hidden automated imports before any script is started, and the `Option Explicit` statement requires to be set before the first import.

Creating a Variable for Captured Data

Capture object information with the **Identify Object** dialog box. You can then create a variable to represent the captured locator information. To do this, capture an object using the **Identify Object** dialog box and paste the object information into the script. Use the `Console.Write` method against the object to retrieve the text held in the object, as shown in the following example:

```
With _desktop.WPFWindow("my main window")
       .WPFTextBox("@automationId='textBoxSingle'").SetText(carmake)
       Console.Write (carmake)
End With
```

Alternatively, the following example also works:

```
<my window
variable>.WPFTextBox("@automationId='textBoxSingle'").SetText(carmake)
Console.Write (carmake)
```

A variable can then be created that represents the result of this statement, which is the text held in the control. Declare the variable using a data type of *String*, and set the variable to the result of the statement. Once the variable is set, its data can be evaluated, validated, displayed, or used for other test purposes. The following shows a variable set to the value of captured text in a message box.

```
Dim carmake As String

carmake = window.WPFTextBox("@automationId='textBoxSingle'").GetLineText(0)

MsgBox (carmake, vbOKOnly)
```

This technique of capturing application information and setting it to a variable can be used on any identifiable object in a test application, such as a window, list box, or button.

In addition to using `Console.Write` method for this purpose, the `Text` property may be used.

The code looks similar to the following:

```
Dim price = browser.DomTextField("@name=txtDealerPrice").Text
```

The `Text` property can return all the data from the control. However the `Text` property is not a member of all Silk Test Workbench objects. Use the `Text` property for supported objects where there is a possibility of data exceeding the size of the display field.

To capture text from more than one control or an entire screen, use `Console.Write`. For instance, to retrieve the complete text of a web site, type the following:

```
Console.WriteLine(_desktop.BrowserApplication().Find("//
HTML").GetProperty("innerText"))
```

For more information about retrieving text for Web sites, see *What is the Difference Between textContents, innerText, and innerHtml?*

Examining Data in Variables

The best way to examine the contents of data variables during playback is to make use of the Visual Basic `MsgBox` function.

Playback causes the contents of the variable to display in a message box. The message box displays until the **OK** button is clicked, at which point playback continues.

The following example shows a simple use of the command to display the contents of a variable:

```
Imports SilkTest.Ntf.Wpf

Public Module Main
    Dim _desktop As Desktop = Agent.Desktop

    Public Sub Main()
        Dim sCarmake = "Honda"
        MsgBox(sCarmake)

        With _desktop.WPFWindow("@caption='WPF Sample Application'")
            .Restore()
            .WPFMenuItem("@caption='Basic Controls'").Select()
        End With
        With _desktop.WPFWindow("@caption='Basic Controls'")
            .WPFTabControl("@automationId='tabControl'").Select("Text")
            .WPFTextBox("@automationId='textBoxSingle'").SetText(sCarmake)
        End With
    End Sub

End Module
```

In the preceding example, the *sCarmake* variable value, which in this case is *Honda*, displays in the message box during playback.

Note: This example is meant to illustrate how you can implement the `MsgBox` function. You must add an application configuration and make any necessary adjustments for your test application to successfully implement this code. If you cut and paste this example into your script, an error occurs because the application configuration is missing.

Using Variables in Scripts

A value contained in double quotes (a string) in any Silk Test Workbench command can be replaced by a value in a variable. Typically this is a string variable. However, you can use variables for other data types such as `Integer` or `Double` when inputting values into a field.

The most common use of variables is as a substitute for a literal value used as input to a control, such as an edit box. Any method which enters data into a control, such as `SetText()` or `TypeKeys()` can substitute a variable for a literal value. To do this, declare the variable, assign its value, then replace the literal value with the variable. For example:

```
browser.DomTextField("@id='name-search:lastName'").SetText("Smith")

'becomes:

Dim lastName = "Smith"

.DomTextField("@id='name-search:lastName'").SetText(sLastName)
```

When selecting an item from a list or menu you can also use a variable for the entry selection. For example:

```
browser.DomListBox("@id='quick-link:jump-menu'").Select("Agent Lookup")

'becomes:

Dim selection = "Auto Quote"

browser.DomListBox("@id='quick-link:jump-menu'").Select(sSelection)
```

Values returned by methods or properties can be placed into a variable for evaluation. A common use of this is to return a value of the Boolean data type for evaluation in conditional logic. The following code

shows how a Boolean value returned by the `AllowsMultiSelect` property is used to evaluate whether a list box control allows selection of multiple items.

```
Dim canMultiSelect As Boolean

' Returns true if the second list box in the window allows multiple selections

canMultiSelect = mainWindow.ListBox("[2]").AllowsMultiSelect

If canMultiSelect Then

    MsgBox ("Employee selection list box allows to select multiple
employees.")

Else

    MsgBox ("Employee selection list box allows to only select one
employees.")

EndIf
```

A variable can replace the raw attach name of a control. Use the string concatenate character &. For example:

```
browser.DomLink("@caption='Eye'").Select()

'could also read:

Dim sLink As String

sLink = "Eye"

browser.DomLink("@caption=" & sLink).Select()
```

Tip: Silk Test Workbench also supports wildcard characters to account for variability in a GUI map or a control or window.

Parameters in scripts can be shared with visual tests, so the same data can be used across a testing solution. For details, see *Passing Data Between Scripts and Visual Tests*.

Passing Data Between Scripts and Visual Tests

Both visual tests and scripts can be used in the same testing solution. Each has certain advantages in an overall testing solution. As such, a testing solution may include both visual tests and scripts. Scripts can be used to perform tasks that are more difficult to perform through visual tests, and vice versa.

To help integrate visual tests and scripts in a testing solution, Silk Test Workbench lets you pass data between scripts and visual tests. This lets you leverage data created in your existing Silk Test Workbench script base, as well as allowing you to use the most efficient testing assets and still retain data integrity within a testing solution. Pass data between scripts and visual tests using script parameters.

Note: To pass data between a visual test and a script, the script must be inserted and played back from within the visual test. To pass data, scripts and visual tests must be in the same project, or the inserted script or visual test must be in a referenced project.

With script parameters, you can:

- Create data in a visual test and use it in a script.
- Create data in a visual test, pass it to a script, modify the data in the script, and pass it back to the visual test.
- Create data in a script and pass it to a visual test.

The process is as follows:

1. Add input and output parameters to a script.
2. Insert the script into the visual test.
3. Pass data from the visual test to the script by editing the value of the input parameter in the visual test.

   Note: The value that you send from the visual test is not saved in the script, but it can be used in computations in the script.
4. In the visual test, create a local variable to store the data passed from the script output parameter, and then use the local variable within the visual test.

Note: For a detailed example of how to create input and output parameters and use them in a visual test, refer to *Playing Back Scripts From Visual Tests* in the *Welcome to the SilkTest Workbench Visual Test Tutorial*.

Defining Parameters to Pass Between a Visual Test and a Script

Any parameters to be passed to and from a script must be defined as input and output parameters for the script.

1. Open the script that you want to use to pass parameters between it and a visual test.
2. Add the input parameters that define the parameters passed from the visual test to the script.
   a) In the **Properties** pane, right-click and choose **Add Input Parameter**. The **Add Script Input Parameter** dialog box opens.
   b) Type a name for the parameter into the **Name** field.
   c) Select a data type for the parameter from the **Type** list.

      Silk Test Workbench supports the following parameter types:

Boolean	The parameter's value is either `True` or `False`.
Number (Double)	The parameter's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Number (Long)	The parameter's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Text	The parameter's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation.

   d) Leave the **Default Value** text box empty.
   e) Click **OK**. The input parameter displays in the list of input parameters in the **Properties** pane.

      Note: Input parameters defined for a script do not appear in the script itself. You can view input parameters in the **Properties** pane.
3. Add the output parameters that define the parameters passed from the script to the visual test.
   a) In the **Properties** pane, right-click and choose **Add Output Parameter**. The **Add Script Output Parameter** dialog box opens.
   b) In the **Name** text box, type the name of the parameter that you want to pass to the visual test.
   c) Click **OK**. The output parameter displays in the list of output parameters in the **Properties** pane.

      Note: Output parameters do not appear in the script unless they are actually declared in the script.

   The following code sample shows how you can use the passed variables. It shows how you can modify the subroutine `Main()` to include the parameters that you want to create and how you can access these parameters using the dictionary. Define *inputParameterName* as an input parameter and *outputParameterName* as an output parameter.

```
Public Module Main
  Dim _desktop As Desktop = Agent.Desktop
```

```
  Public Sub Main(args As IDictionary(Of String, Object))
    Dim text As String = args.Item("inputParameterName")
    args.Item("outputParameterName") = "someValue"
  End Sub
End Module
```

4. Open the visual test that you want to use to pass parameters between it and a script.
5. If the script has been included in at least one visual test, to view the most up-to-date parameters in the visual test, in the **Properties** pane of the inserted script step, click the **Name** text box, and then click **Refresh Parameters**.
6. Edit the value of the input parameter that you want to use in the script in the **Properties** pane in the visual test. When you playback the visual test, the input parameter is passed to the script.

Note: The value that you send from the visual test is not saved in the script, but it can be used in computations in the script.

Output parameters must also be set as local variables in a visual test to pass the parameters to the visual test.

Executing a Script Within a Visual Test

Visual tests can call and execute existing .NET scripts. This allows a modularized approach to application testing and provides greater control over test execution. When executing a script within a visual test, it is important to remember to keep test site integrity and ensure any test applications being tested are in the correct initial playback state.

You can also use this procedure to create a driver test.

1. Open the visual test in which you want to insert the script.
2. Select the test step that precedes where you want the inserted script to execute.
3. Choose **Insert** > **.NET Script**. The **Browse for .NET Script** dialog box opens.
4. Select the script to insert and then click **OK**. Silk Test Workbench inserts a step below the selected step. The inserted step calls and plays back the selected script. The step text is as follows:

   ```
   Playback .NET Script 'asset name'
   ```

   where 'asset name' is the name of the inserted script.

During playback, when the preceding step executes, the inserted script plays back to completion before the next step in the visual test executes. After a script has been inserted into a visual test, you can edit it and configure the settings used to play it back.

The tested applications are configured, hooked, and started when required, and remain hooked until the end of the playback session. For example, if the visual test has an application configuration for Notepad, and the .NET script has an application configuration for Calculator, the following happens:

- Notepad is hooked and started when the execution of the visual test starts.
- Calculator is hooked and started when the execution of the inserted .NET script starts. Notepad remains hooked.
- Both Notepad and Calculator remain hooked when the execution of the inserted .NET script stops.
- Both Notepad and Calculator get unhooked when the playback session ends, which means that the execution of the visual test stops.

Using Parameters Passed From a Script in a Visual Test

Once input and output parameters have been defined for a script, the output parameters can be passed to a visual test that executes the script during its playback. However, to use the passed parameters in the visual test, they must be set to local variables that are stored in the visual test.

1. Open the visual test.
2. In the **Test Steps** pane, select the step in the visual test that executes the script. The step text looks similar to `Playback .NET Script 'asset name'`, where *asset name* is the name of the inserted script.

   Properties for the step display in the **Properties** pane. This includes any input and output parameters for the script.
3. In the **Properties** pane for the step, expand the **Output parameters** category.

   The names of variables defined as output parameters to be passed from the script to the visual test appear in the Output parameters subcategories. These parameters must now be set to local variables created for the visual test.

   If the parameters do not appear in the **Properties** pane, reload the script that contains the parameters. To do this, in the **Properties** pane of the inserted script step, click the **Name** text box, and then click **Update Parameter Names**.
4. Click the text box of an output parameter name in the **Properties** pane and select the appropriate local variable from the list. This sets the value of the parameter passed from the script to a local variable in the visual test so it can be used in the visual test.
5. Repeat the preceding step for each output parameter in the **Properties** pane to be passed to from the script to the visual test.

   Note: To see the values passed from the script to a visual test, insert result comment steps into the visual test. For the comment Expression property, click **Select** and then choose **Variable**. Select a local variable from the list that contains the passed value.
6. In the visual test, modify the appropriate test step to include the local variable.

   The local variable passes the data from the script output parameter to the test step.

   a) Select the step that you want to use in the **Test Steps** pane.
   b) In the **Properties** pane, click the text box for the property that corresponds with the local variable property type.
      For example, you might click the **text** text box in the **Parameters** category.
   c) Click the **Select** button, and then select **Variable**. The **Select a Variable** dialog box opens.
   d) Select the local variable that you want to use, and then click **OK**. The step text displays as:`Enter '[`*localvariablename*`]'`.

If the script has been included for playback in the visual test and its input and output parameters have been defined correctly, the visual test can now be played back and parameters are successfully passed between the visual test and the script within the visual test.

Note: For a detailed example of how to create input and output parameters and use them in a visual test, refer to *Playing Back Scripts From Visual Tests* in the *Welcome to the SilkTest Workbench Visual Test Tutorial*.

Creating and Passing Parameters in Scripts

Creating and passing parameters is helpful in a team testing environment, where an experienced tester can create a library of scripts that perform common testing functions from which a novice developer can select from when creating a more basic script. Creating and passing parameters also works well in single tester environments where complex functions are reused in several scripts.

You can create and pass parameters between a parent and child script or using a single script. The advantage of using a parent and child scenario is that multiple child scripts can use the parent script. For example, if the parent script creates a random number, you might want to use that functionality in several child scripts.

Creating and Passing Parameters Between Scripts

You can create and pass parameters between a parent and child script or using a single script. The advantage of using a parent and child scenario is that multiple child scripts can use the parent script. For

example, if the parent script creates a random number, you might want to use that functionality in several child scripts.

1. Create a parent script that includes the parameters that you want to pass.
   a) Choose **File** > **New**. The **New Asset** dialog box opens.
   b) Select **.NET Script** from the asset types list, and then type a name for the script in the **Asset name** text box.

      For example, you might name the script "`NameParameters`."
   c) Click **OK** to save the script. Silk Test Workbench saves the script as an asset and displays the script template.
   d) Modify the `Main()` sub to include the parameters that you want to create and pass to child scripts. For example, the following parent script passes a string for the first name and last name in the test application.

```
Public Sub Main()
            Dim args As New Dictionary(Of String, Object)
            args("FName") = "Chris"
            args("LName") = "Smith"
```

   e) Include the `RunScript` command that calls the child script and specifies the name for the parameters that you created.
      For example, the following parent script passes a string for the first name and last name in the test application, and calls the child script named *childscript* with the parameter called *args*.

```
Public Module Main
      Dim _desktop As Desktop = Agent.Desktop

      Public Sub Main()
            Dim args As New Dictionary(Of String, Object)
            args("FName") = "Chris"
            args("LName") = "Smith"

            Workbench.RunScript("childscript", args)

      End Sub
End Module
```

2. *Optional*: To include a message box that returns the parameters that the script enters, add the following code to the parent script:

```
MsgBox("Hello, " + args("FName") + "!")
MsgBox ("Hello, " + args("FName") + " " +  args("LName") + "!")
```

   A message box opens when the script plays back and shows the values that the test enters.

3. Create a child script that calls the parameters from the parent script.
   For example, to work with the first name and last name in the sample Web application Sign Up form, record a script for the Sign Up form that includes a first name and last name. The script might look like the following example:

```
Imports SilkTest.Ntf.XBrowser
Public Module Main
    Dim _desktop As Desktop = Agent.Desktop

    Public Sub Main()
        With _desktop.BrowserWindow("/BrowserApplication[1]//BrowserWindow")
            .DomButton("@id='login-form:signup'").Select()
            .DomTextField("@id='signup:fname'").SetText("first")
            .DomTextField("@id='signup:lname'").SetText("last")
      .DomElement("@src='http://extjs.com/
s.gif'").DomClick(MouseButton.Left,
     New Point(8, 16))
            .DomButton("@textContents='Today'").Select()
            .DomTextField("@id='signup:email'").SetText("test@test1.com")
            .DomTextField("@id='signup:street'").SetText("123 street rd")
```

```
            .DomTextField("@id='signup:city'").SetText("Marlton")
            .DomListBox("@id='signup:state'").Select("Massachusetts")
            .DomTextField("@id='signup:zip'").SetText("09876")
            .DomTextField("@id='signup:password'").SetText("test")
            .DomButton("@id='signup:signup'").Select()
            .DomButton("@id='signup:continue'").Select()
        End With

    End Sub
End Module
```

4. Modify the `Sub Main()` to take one parameter that is an `IDictionary(Of String, Object)` that you want to set and pass to parent scripts.

   The `IDictionary` class is provided by .NET. You can review the documentation for the class at *http://msdn.microsoft.com/en-us/library/s4ys34ea.aspx*.

   The revised `Sub Main ()` code will look like the following:

   ```
   Public Sub Main(args As IDictionary(Of String, Object))
   ```

5. Modify the controls that you want to use parameters.

   For example, to modify the first and last name in the Web application recording, replace the following code:

   ```
   .DomTextField("@id='signup:fname'").SetText("first")
   .DomTextField("@id='signup:lname'").SetText("last")
   ```

   with:

   ```
   .DomTextField("@id='signup:fname'").SetText(args("FName"))
   .DomTextField("@id='signup:lname'").SetText(args("LName"))
   ```

6. From the parent script, click **Playback**.

   Note: If you run a script with parameters from the child script, it fails since no parameters are specified. However, you can add a second `Main()` sub with no parameters, and pass in default parameter values to the other `Main()` to run from a single script directly.

Creating and Passing Parameters Within a Script

You can create and pass parameters between a parent and child script or using a single script. The advantage of using a parent and child scenario is that multiple child scripts can use the parent script. For example, if the parent script creates a random number, you might want to use that functionality in several child scripts.

1. Create a script.
   a) Choose **File** > **New**. The **New Asset** dialog box opens.
   b) Select **.NET Script** from the asset types list, and then type a name for the script in the **Asset name** text box.

      For example, you might name the script "`NameParameters`."
   c) Click **OK** to save the script. Silk Test Workbench saves the script as an asset and displays the script template.
2. Record or type the script that tests the application.
   For example, to work with the first name and last name in the sample Web application Sign Up form, record a script for the Sign Up form that includes a first name and last name. The script might look like the following example:

   ```
   Imports SilkTest.Ntf.XBrowser
   Public Module Main
     Dim _desktop As Desktop = Agent.Desktop

     Public Sub Main()
       With _desktop.BrowserWindow("/BrowserApplication[1]//BrowserWindow")
         .DomButton("@id='login-form:signup'").Select()
   ```

```
      .DomTextField("@id='signup:fname'").SetText("first")
      .DomTextField("@id='signup:lname'").SetText("last")
      .DomElement("@src='http://extjs.com/
s.gif'").DomClick(MouseButton.Left,
        New Point(8, 16))
      .DomButton("@textContents='Today'").Select()
      .DomTextField("@id='signup:email'").SetText("test@test1.com")
      .DomTextField("@id='signup:street'").SetText("123 street rd")
      .DomTextField("@id='signup:city'").SetText("Marlton")
      .DomListBox("@id='signup:state'").Select("Massachusetts")
      .DomTextField("@id='signup:zip'").SetText("09876")
      .DomTextField("@id='signup:password'").SetText("test")
      .DomButton("@id='signup:signup'").Select()
      .DomButton("@id='signup:continue'").Select()
    End With

  End Sub
End Module
```

3. Create a second `Main()` sub to include the parameters that you want to create and pass to child scripts.
   For example, the following script passes a string for the first name and last name in the test application.

```
Public Sub Main()
  Dim args As New Dictionary(Of String, Object)
  args("FName") = "Chris"
  args("LName") = "Smith"
```

4. Include the `Main(args)` command that calls the first `Main()` sub script and specifies the name for the parameters that you created.
   For example, the following code passes a string for the first name and last name in the test application, and calls the parameter called *args*.

```
Public Module Main
  Dim _desktop As Desktop = Agent.Desktop

  Public Sub Main()
    Dim args As New Dictionary(Of String, Object)
    args("FName") = "Chris"
    args("LName") = "Smith"

    Main(args)

  End Sub
End Module
```

5. Modify the `Sub Main()` that tests the application to take one parameter that is an `IDictionary(Of String, Object)` that you want to set and pass to parent scripts.

   The `IDictionary` class is provided by .NET. You can review the documentation for the class at *http://msdn.microsoft.com/en-us/library/s4ys34ea.aspx*.

   The revised `Sub Main ()` code looks like the following:

```
Public Sub Main(args As IDictionary(Of String, Object))
```

6. Modify the controls that you want to use parameters.

   For example, to modify the first and last name in the Web application recording, replace the following code:

```
.DomTextField("@id='signup:fname'").SetText("first")
.DomTextField("@id='signup:lname'").SetText("last")
```

   with:

```
.DomTextField("@id='signup:fname'").SetText(args("FName"))
.DomTextField("@id='signup:lname'").SetText(args("LName"))
```

The script looks like the following:

```
Imports SilkTest.Ntf.XBrowser
Public Module Main
  Dim _desktop As Desktop = Agent.Desktop

  Public Sub Main(args As IDictionary(Of String, Object))
  With _desktop.BrowserWindow("/BrowserApplication[1]//BrowserWindow")
    .DomButton("@id='login-form:signup'").Select()
    .DomTextField("@id='signup:fname'").SetText(args("FName"))
    .DomTextField("@id='signup:lname'").SetText(args("LName"))
    .DomElement("@src='http://extjs.com/s.gif'").DomClick(MouseButton.Left,
      New Point(8, 16))
    .DomButton("@textContents='Today'").Select()
    .DomTextField("@id='signup:email'").SetText("test@test1.com")
    .DomTextField("@id='signup:street'").SetText("123 street rd")
    .DomTextField("@id='signup:city'").SetText("Marlton")
    .DomListBox("@id='signup:state'").Select("Massachusetts")
    .DomTextField("@id='signup:zip'").SetText("09876")
    .DomTextField("@id='signup:password'").SetText("test")
    .DomButton("@id='signup:signup'").Select()
    .DomButton("@id='signup:continue'").Select()
  End With

  End Sub

  Public Sub Main()
    Dim args As New Dictionary(Of String, Object)
    args("FName") = "Chris"
    args("LName") = "Smith"

    Main(args)

  End Sub

End Module
```

7. *Optional*: To include a message box that returns the parameters that the script enters, add the following code:

```
MsgBox("Hello, " + args("FName") + "!")
MsgBox ("Hello, " + args("FName") + " " +  args("LName") + "!")
```

A message box opens when the script plays back and shows the values that the test enters.

The script looks like the following:

```
Imports SilkTest.Ntf.XBrowser
Public Module Main
  Dim _desktop As Desktop = Agent.Desktop

  Public Sub Main(args As IDictionary(Of String, Object))
  With _desktop.BrowserWindow("/BrowserApplication[1]//BrowserWindow")
    .DomButton("@id='login-form:signup'").Select()
    .DomTextField("@id='signup:fname'").SetText(args("FName"))
    .DomTextField("@id='signup:lname'").SetText(args("LName"))
    .DomElement("@src='http://extjs.com/s.gif'").DomClick(MouseButton.Left,
      New Point(8, 16))
    .DomButton("@textContents='Today'").Select()
    .DomTextField("@id='signup:email'").SetText("test@test1.com")
    .DomTextField("@id='signup:street'").SetText("123 street rd")
    .DomTextField("@id='signup:city'").SetText("Marlton")
    .DomListBox("@id='signup:state'").Select("Massachusetts")
    .DomTextField("@id='signup:zip'").SetText("09876")
    .DomTextField("@id='signup:password'").SetText("test")
    .DomButton("@id='signup:signup'").Select()
```

```
    .DomButton("@id='signup:continue'").Select()
  End With

  End Sub

  Public Sub Main()
    Dim args As New Dictionary(Of String, Object)
    args("FName") = "Chris"
    args("LName") = "Smith"

    Main(args)

    MsgBox("Hello, " + args("FName") + "!")
    MsgBox ("Hello, " + args("FName") + " " +  args("LName") + "!")

  End Sub

End Module
```

8. Click **Playback**.

Examples of Scripts that Contain Parameters

The following examples show scripts that create and pass parameters.

List Example

The following example demonstrates how to create and pass list parameters between a parent and child script that test the Notepad application.

The parent script contains:

```
Public Module Main
 Dim _desktop As Desktop = Agent.Desktop

  Public Sub Main()
    Dim numList As New List(Of Integer)
    For i As Integer = 0 To 10
    numList.Add(i)
    Next

    Dim args As New Dictionary(Of String, Object)
    args("Numbers") = numList

    Workbench.RunScript("ChildScriptArgs", args)
  End Sub
End Module
```

The child script includes:

```
Public Module Main
  Dim _desktop As Desktop = Agent.Desktop
  Public Sub Main(args As IDictionary(Of String, Object))
    Dim nums As List(Of Integer) = args("Numbers")

    With _desktop.Window("/Window[@caption='Untitled - Notepad']")
    For Each num As Integer In nums
    .TextField().TypeKeys(num.ToString() + "<Enter>")
    Next
    End With

  End Sub
End Module
```

Enum Example

The following example demonstrates how to create and pass enum parameters between a parent and child script that test the Notepad application.

The parent script contains:

```
Public Module Main
 Dim _desktop As Desktop = Agent.Desktop

  Public Sub Main()
    Dim args As New Dictionary(Of String, Object)
    args("Day") = DayOfWeek.Friday

    Workbench.RunScript("ChildDayScript", args)

  End Sub
End Module
```

The child script includes:

```
Public Module Main
  Dim _desktop As Desktop = Agent.Desktop

  Public Sub Main(args As IDictionary(Of String, Object))
    Dim day As DayOfWeek = args("Day")

    With _desktop.Window("/Window[@caption='Untitled - Notepad']")
    .TextField().TypeKeys("The day of the week is " + day.ToString() + ".")
    End With

  End Sub
End Module
```

Number Example

The following example demonstrates how to create and pass number parameters in a single script that tests the Notepad application.

The script contains:

```
Public Module Main
 Dim _desktop As Desktop = Agent.Desktop

  Public Sub Main(args As IDictionary(Of String, Object))
    Dim N1 As Integer = CType(args("Num1"), Integer)
    Dim N2 As Integer = CType(args("Num2"), Integer)

      With _desktop.Window("@caption='Untitled - Notepad'")
        .TextField().ClearText()
        .TextField().TypeKeys("<Enter>"+ "Numbers" + "<Enter>")
        .TextField().TypeKeys(N1)
        .TextField().TypeKeys("<Enter>")
        .TextField().TypeKeys(N2)
        .TextField().TypeKeys("<Enter>")
      End With
  End Sub

  Public Sub Main()
    Dim args As New Dictionary(Of String, Object)

      args.Add("Num1", 35)
      args.Add("Num2", 25)

      Main(args)
```

```
  End Sub
End Module
```

Boolean Example

The following example demonstrates how to create and pass boolean parameters in a single script that tests the Notepad application.

The script contains:

```
Public Module Main
 Dim _desktop As Desktop = Agent.Desktop

  Public Sub Main(args As IDictionary(Of String, Object))
    Dim B1 As Boolean = CType(args("Bool1"), Boolean)
    Dim B2 As Boolean = CType(args("Bool2"), Boolean)

    With _desktop.Window("@caption='Untitled - Notepad'")
      .TextField().ClearText()
      .TextField().TypeKeys("<Enter>" +"Boolean Values" + "<Enter>")
      .TextField().TypeKeys(B1)
      .TextField().TypeKeys("<Enter>")
      .TextField().TypeKeys(B2)
      .TextField().TypeKeys("<Enter>")
    End With
  End Sub

  Public Sub Main()
  Dim args As New Dictionary(Of String, Object)

    args.Add("Bool1", True)
    args.Add("Bool2", False)

    Main(args)
  End Sub
End Module
```

Array Example

The following example demonstrates how to create and pass array parameters in a single script that tests the Notepad application.

The script contains:

```
Public Module Main
 Dim _desktop As Desktop = Agent.Desktop

  Public Sub Main(args As IDictionary(Of String, Object))
    Dim A1() As Integer = CType(args("Arr1"), Integer())

      With _desktop.Window("@caption='Untitled - Notepad'")

      .TextField().TypeKeys("Array Numbers<Enter>")

      For Each num As Integer In A1
      .TextField().TypeKeys(num)
      .TextField().TypeKeys("<Enter>")
      Next
      .TextField().TypeKeys("<Enter>")

      End With
  End Sub

  Public Sub Main()
    Dim args As New Dictionary(Of String, Object)
```

```
    args("Arr1") = New Integer() {1,2,3,4,5}

    Main(args)
  End Sub
End Module
```

Managing Scripts

This section describes how you can manage VB .NET scripts.

Finding and Replacing Code in Scripts

Find and replace code in VB .NET scripts to speed script modification.

1. Open the script that you want to search.
2. Choose **Edit** > **Find and Replace** and then one of the following options:
   - **Find** – In the **Find what** text box, specify the code that you want to find and then click **Find Next**.
   - **Find Next** – Choose this option to find the next instance of the item you specified in the **Find what** text box.
   - **Find Previous** – Choose this option to find the item you specified in the **Find what** text box in the script that precedes the last entry that was found.
   - **Replace** – In the **Find what** text box, specify the code that you want to find. In the **Replace with** text box, specify the code that you want to replace the existing code with and then click **Replace** or **Replace All**.

Navigating Code in Scripts Using Bookmarks

Use bookmarks to navigate to key portions of the script.

1. Open the script in which you want to include bookmarks.
2. Choose **Edit** > **Bookmarks** and then one of the following options:
   - **Toggle Bookmark** – Choose this option to include or hide a bookmark on the selected line of code.
   - **Next Bookmark** – Choose this option to go to the next bookmark in the script.
   - **Previous Bookmark** – Choose this option to go to the bookmark in the script that precedes the last bookmark that was found.
   - **Clear Bookmarks** – Choose this option to remove all bookmarks from the script.

Adding Script Input Parameters

Scripts can receive data from a visual test in an input parameter, and, conversely, pass data to visual tests in an output parameter.

1. In the **Properties** pane, right-click and choose **Add Input Parameter**. The **Add Script Input Parameter** dialog box opens.
2. Type a name for the parameter into the **Name** field.

   Valid characters for parameter names include uppercase and lowercase alphanumeric characters and the underscore ("ABC", "abc", "_"). If a parameter name being added or edited is a duplicate of another parameter name, supply a unique name for the parameter.
3. Select a data type for the parameter from the **Type** list.

   Silk Test Workbench supports the following parameter types:

   Boolean The parameter's value is either `True` or `False`.

Number (Double)	The parameter's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Number (Long)	The parameter's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Text	The parameter's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation.

4. Leave the **Default Value** text box empty.

   If left empty, the initial value of the parameter as set in the visual test is used. The value can be changed in a visual test or script if passed to a script. The value must be consistent with the type. For example, if the type is Number (Long), the initial value must be an integer.
5. Click **OK**. The input parameter displays in the list of input parameters in the **Properties** pane.

Editing Script Input Parameters

Edit an input parameter in a script to reflect possible changes in the script or visual test that passes the parameter.

1. In the **Properties** pane, right-click the input parameter that you want to edit and choose **Edit *<inputparametername>***. The **Edit Script Input Parameter** dialog box opens.
2. Type a name for the parameter into the **Name** field.

   Valid characters for parameter names include uppercase and lowercase alphanumeric characters and the underscore ("ABC", "abc", "_"). If a parameter name being added or edited is a duplicate of another parameter name, supply a unique name for the parameter.
3. Select a data type for the parameter from the **Type** list.

   Silk Test Workbench supports the following parameter types:

Boolean	The parameter's value is either `True` or `False`.
Number (Double)	The parameter's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Number (Long)	The parameter's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Text	The parameter's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation.

4. Leave the **Default Value** text box empty.

   If left empty, the initial value of the parameter as set in the visual test is used. The value can be changed in a visual test or script if passed to a script. The value must be consistent with the type. For example, if the type is Number (Long), the initial value must be an integer.
5. Click **OK**.

Deleting an Input Parameter from a Script

If necessary, you can delete an input parameter from a script.

1. In the **Properties** pane, right-click the input parameter that you want to delete and choose **Delete *<inputparametername>***.

2. Click **Yes** to confirm that you want to delete the input parameter.

Adding Script Output Parameters

Scripts can receive data from a visual test in an input parameter, and, conversely, pass data to visual tests in an output parameter.

1. In the **Properties** pane, right-click and choose **Add Output Parameter**. The **Add Script Output Parameter** dialog box opens.
2. In the **Name** text box, type the name of the parameter that you want to pass to the visual test.

   Valid characters for parameter names include uppercase and lowercase alphanumeric characters and the underscore ("ABC", "abc", "_"). If a parameter name being added or edited is a duplicate of another parameter name, supply a unique name for the parameter.
3. From the **Type** list, select the data type of the local variable in the visual test for which you want to pass data.

   Silk Test Workbench supports the following parameter types:

Boolean	The parameter's value is either `True` or `False`.
Number (Double)	The parameter's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Number (Long)	The parameter's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Text	The parameter's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation.

4. Click **OK**. The output parameter displays in the list of output parameters in the **Properties** pane.

Editing Script Output Parameters

Edit an output parameter of a script to pass different data to a visual test.

1. In the **Properties** pane, right-click the output parameter that you want to edit and choose **Edit *<outputparametername>***.
2. In the **Name** text box, type the name of the parameter that you want to pass to the visual test.

   Valid characters for parameter names include uppercase and lowercase alphanumeric characters and the underscore ("ABC", "abc", "_"). If a parameter name being added or edited is a duplicate of another parameter name, supply a unique name for the parameter.
3. From the **Type** list, select the data type of the local variable in the visual test for which you want to pass data.

   Silk Test Workbench supports the following parameter types:

Boolean	The parameter's value is either `True` or `False`.
Number (Double)	The parameter's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Number (Long)	The parameter's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.

Text	The parameter's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation.

4. Click **OK**.

Deleting an Output Parameter from a Script

If necessary, you can delete an output parameter from a script.

1. In the **Properties** pane, right-click the output parameter that you want to delete and choose **Delete *<outputparametername>***.
2. Click **Yes** to confirm that you want to delete the output parameter.

Adding a File to a Script

You can add a file that the script references to logically segment code for distinct purposes. For example, if your script requires 4000 lines of code, you might want to include only core lines of code in the **Main** window and create separate files for helper code. Helper code might include code that connects the main application to a database for data transfer purposes.

1. To add a file to a script, perform one of the following steps:
   - Choose **Insert** > **File**.
   - Right-click in the **Properties** pane and then choose **Add File**.

   A new tab opens in the **Code** window.
2. Add the code that you want to include in the file and then click **Save**.
3. Click the **Main** tab and include a reference to the file.

Deleting a File from a Script

You can delete a file that a script references.

1. Open the script that contains the file that you want to delete.
2. In the **Properties** pane, expand the **Files** category, right-click the file name and then choose **Delete File**.

Renaming a File in a Script

You can rename a file that a script references.

1. Open the script that contains the file that you want to rename.
2. In the **Properties** pane, expand the **Files** category, right-click the file name and then choose **Rename File**.
3. Type the new name in the **Properties** pane.

Referencing .NET Assemblies from a .NET Script

When you record a script, all the references that are appropriate for the controls that you interacted with are automatically added to the reference library. If you manually create a script or add an additional application configuration, you might need to manually add references to the library. You can add .NET assemblies from the .NET framework as references for a .NET script.

1. To add a reference to the script reference library, right-click in the **Properties** pane and then choose **Add .NET Assembly Reference**. The **Add .NET Assembly Reference** dialog box opens.
2. To add a component, perform one of the following steps:
   - Double-click a component from the component list.

- Click **Browse** and navigate to the component that you want to add, click **Open**, and then click **OK**.

The **Path** column lists the location of the DLL file.

The new component displays in the **Properties** pane in the **References** category.

Referencing Classes in Another .NET Script from a .NET Script

To use classes defined in another .NET script in a .NET script, add a reference to the other .NET script to the .NET script.

1. Right-click in the **Properties** pane and then choose **Add .NET Script Reference**. The **Browse for .NET Script** dialog box opens.
2. Select which .NET scripts the dialog box should display:
   - To display all .NET scripts created by you, click **By me**.
   - To display all .NET scripts created by all users, click **By all users**.
3. To add a .NET script from the list as a reference to the current .NET script, perform one of the following steps:
   - Double-click a .NET script from the list.
   - Select a .NET script in the list and click **OK**.
   - Select a .NET script in the list and click **Insert & Open** to add the .NET script as a reference and to open the referenced script.

The referenced .NET script displays in the **References** category of the **Properties** pane.

Removing a Reference from the Script Reference Library

To remove a reference from the script reference library, right-click in the **Properties** pane and then choose **Remove Reference**.

Browse for .NET Script Dialog Box

Use the **Browse for .NET Script** dialog box to quickly find and select scripts previously created by you or by other users.

Note: Only scripts from the *Common* project display in the **Select an Asset** list.

The **Browse for .NET Script** dialog box contains the following controls:

Control	Description
Show assets created	A list box that allows you to define which assets you want to display in the **Select an asset** list. Choose **By me** to display only assets created by you, or select **By all users** to display assets created by any user.
Name Filter	A text field into which you can type the name of an asset, to display only assets which match this name in the **Select an asset** list.
Select an asset	A list of assets which takes any applied filters into account.

Select the desired VB .NET script from the **Select an asset** list, then click **OK** to use it.

Note: Click any column header to alphabetically sort the **Select an asset** list by the column type.

Calling Windows DLLs

This section describes how you can call DLLs. You can call a DLL either within the process of the Open Agent or in the application under test (AUT). This allows the reuse of existing native DLLs in test scripts.

DLL calls in the Open Agent are typically used to call global functions that do not interact with UI controls in the AUT.

DLL calls in the AUT are typically used to call functions that interact with UI controls of the application. This allows Silk Test Workbench to automatically synchronize the DLL call during playback.

Note: In 32-bit applications, you can call 32-bit DLLs, while in 64-bit applications you can call 64-bit DLLs. The Open Agent can execute both 32-bit and 64-bit DLLs.

Note: The .NET framework also provides built-in support for DLL calling, which is called P/Invoke. P/Invoke can be used in Visual Basic scripts to call DLL functions within the process that executes the script. However, in contrast to calling DLL functions with Silk Test Workbench in the application under test, there is no automatic synchronization.

Note: You can only call DLLs with a C interface. If you want to call .NET assemblies, which also have the file extension .dll, do not use the DLL calling feature but instead just add a reference to the assembly in your .NET script.

Calling a Windows DLL from Within a Script

A declaration for a DLL starts with an interface that has a Dll attribute. The syntax of the declaration is the following:

```
<Dll("dllname.dll")> Public Interface DllInterfaceName
  FunctionDeclaration
  [FunctionDeclaration]…
End Interface
```

dllname	The name of or the full path to the DLL file that contains the functions you want to call from your Visual Basic scripts. Environment variables in the DLL path are automatically resolved. You do not have to use double backslashes (\\) in the path, single backslashes (\) are sufficient.
DllInterfaceName	The identifier that is used to interact with the DLL in a script.
FunctionDeclaration	A function declaration of a DLL function you want to call.

DLL Function Declaration Syntax

A function declaration for a DLL typically has the following form:

```
Function function-name( [arg-list] )  As return-type
```

For functions that do not have a return value, the declaration has the following form:

```
Sub function-name( [arg-list] )
```

return-type	The data type of the return value.
function-name	The name of the function.
arg-list	A list of the arguments that are passed to the function. The list is specified as follows: `[pass-mode] identifier As data-type`

pass-mode Specifies whether the argument is passed into the function (ByVal) or if it can be modified by the function (ByRef). You can also use the ByRef keyword to pass the value of the argument into the function, where the function changes the value of the argument and passes the new value out.

data-type The data type of the argument.

identifier The name of the argument.

DLL Calling Example

This example writes the text *hello world!* into a field by calling the `SendMessage` DLL function from `user32.dll`.

DLL Declaration:

```
// VB .NET code
<Dll("user32.dll")> Public Interface IUserDll32Functions
  Function SendMessageW( _
    ByVal obj As TestObject,ByVal message As Integer,ByVal wParam As
Integer,ByVal lParam As String) As Integer
End Interface
```

The following code shows how to call the declared DLL function in the AUT:

```
// VB .NET code
Public Sub Main()
  Dim user32Functions As IUserDll32Functions =
DllCall.CreateInProcessDllCall(Of IUserDll32Functions)()
  Dim textField = _desktop.Window().TextField()
  user32Functions.SendMessageW(textField, WindowsMessages.WM_SETTEXT, 0, "my
text")
End Sub
```

Note: You can only call DLL functions in the AUT if the first parameter of the DLL function has the C data type HWND.

The following code shows how to call the declared DLL functions in the process of the Open Agent:

```
// VB .NET code
Public Sub Main()
  Dim user32Functions As IUserDll32Functions = DllCall.CreateAgentDllCall(Of
IUserDll32Functions)()
  Dim textField = _desktop.Window().TextField()
  user32Functions.SendMessageW(textField, WindowsMessages.WM_SETTEXT, 0, "my
text")
End Sub
```

Note: The example code uses the `WindowsMessages` class that contains useful constants for usage with DLL functions that relate to Windows messaging.

Passing Arguments to DLL Functions

DLL functions are written in C, so the arguments that you pass to these functions must have the appropriate C data types. The following data types are supported:

Integer Use this data type for arguments or return values with the following data types:

- int
- INT
- long

- LONG
- DWORD
- BOOL
- WPARAM
- HWND

The Visual Basic type Integer works for all DLL arguments that have a 4-byte value.

Long	Use this data type for arguments or return values with the C data types long and int64. The Visual Basic type Long works for all DLL arguments that have an 8-byte value.
Short	Use this data type for arguments or return values with the C data types short and WORD. The Visual Basic type Short works for all DLL arguments that have a 2-byte value.
Boolean	Use this data type for arguments or return values with the C data type bool.
String	Use this for arguments or return values that are Strings in C.
Double	Use this for arguments or return values with the C data type double.
Single	Use this for arguments or return values with the C data type float.
SilkTest.Ntf.Rectangle	Use this for arguments with the C data type RECT. Rectangle cannot be used as a return value.
SilkTest.Ntf.Point	Use this for arguments with the C data type POINT. Point cannot be used as a return value.
SilkTest.Ntf.TestObject	Use this for arguments with the C data type HWND. TestObject cannot be used as a return value, however you can declare DLL functions that return a HWND with an Integer as the return type.  **Note:** The passed TestObject must implement the SilkTest.Ntf.INativeWindow interface so that Silk Test Workbench is able to determine the window handle for the TestObject that should be passed into the DLL function. Otherwise an exception is thrown when calling the DLL function.
List	Use this for arrays for user defined C structs. Lists cannot be used as a return value.  **Note:** When you use a List as an output parameter, by using ByRef, the list that is passed in must be large enough to hold the returned contents.  **Note:** A C struct can be represented by a List, where every list element corresponds to a struct member. The first struct member is represented by the first element in the list, the second struct members is represented by the second element in the list, and so on.

 **Note:** Any argument that you pass to a DLL function must have one of the preceding Visual Basic data types.

Passing Arguments that Can Be Modified by the DLL Function

An argument whose value will be modified by a DLL function needs to be declared using the ByRef keyword.

Example

This example uses the `GetCursorPos` function of the `user32.dll` in order to retrieve the current cursor position.

DLL declaration:

```
<Dll( "user32.dll" )> Public Interface IUserDll32Functions
  Function GetCursorPos( ByRef point As Point) As Integer
End Interface
```

Usage:

```
Public Sub Main()
  Dim user32Functions As IUserDll32Functions =
DllCall.CreateAgentDllCall( Of IUserDll32Functions)()
  Dim point As Point
  user32Functions.GetCursorPos(point)
  MsgBox( "cursor position = (" & point.X & ", " & point.Y &
")" )
End Sub
```

Passing String Arguments to DLL Functions

Strings that are passing into a DLL function or that are returned by a DLL function are treated by default as Unicode Strings. If your DLL function requires ANSI String arguments, use the `CharacterSet` property of the `DllFunctionOptions` attribute.

Example

```
<Dll( "user32.dll" )> Public Interface IUserDll32Functions
  <DllFunctionOptions(CharacterSet:=CharacterSet.Ansi)>
Function SendMessageA( _
    ByVal obj As TestObject, ByVal message As Integer , ByVal
wParam As Integer , ByRef lParam As String ) As Integer
End Interface
```

Passing a String back from a DLL call as a ByRef argument works per default if the String's size does not exceed 256 characters length. If the String that should be passed back is longer than 256 characters, you need to pass a Visual Basic String in that is long enough to hold the resulting String.

Example

Use the following code to create a String with 1024 blank characters:

```
Dim longEmptyString = New String ( " "c , 1024 )
```

Pass this String as a ByRef argument into a DLL function and the DLL function will pass back Strings of up to 1024 characters of length.

When passing a String back from a DLL call as a function return value, the DLL should implement a DLL function called `FreeDllMemory` that accepts the C String pointer returned by the DLL function and that frees the previously allocated memory. If no such function exists the memory will be leaked.

Aliasing a DLL Name

If a DLL function has the same name as a reserved word in Visual Basic, or the function does not have a name but an ordinal number, you need to rename the function within your declaration and use the alias statement to map the declared name to the actual name.

Example

For example, the `Exit` statement is reserved by the Visual Basic compiler. Therefore, to call a function named `exit`, you need to declare it with another name, and add an alias statement, as shown here:

```
<Dll("mydll.dll")> Public Interface IMyDllFunctions
  <DllFunctionOptions(Alias:="exit")> Sub MyExit()
End Interface
```

Conventions for Calling DLL Functions

The following calling conventions are supported when calling DLL functions:

- __stdcall
- __cdecl

The *__stdcall* calling convention is used by default when calling DLL functions. This calling convention is used by all Windows API DLL functions.

You can change the calling convention for a DLL function by using the `CallingConvention` property of the `DllFunctionOptions` attribute.

Example

The following code example declares a DLL function with the *__decl* calling convention:

```
<Dll("msvcrt.dll")> Public Interface IMsVisualCRuntime

<DllFunctionOptions(CallingConvention:=CallingConvention.Cdecl)>
 Function cos(ByVal input As Double) As Double
End Interface
```

Improving Object Recognition with Microsoft Accessibility

You can use Microsoft Accessibility (Accessibility) to ease the recognition of objects at the class level. There are several objects in Internet Explorer and in Microsoft applications that Silk Test Workbench can better recognize if you enable Accessibility. For example, without enabling Accessibility Silk Test Workbench records only basic information about the menu bar in Microsoft Word and the tabs that appear. However, with Accessibility enabled, Silk Test Workbench fully recognizes those objects.

Example

Without using Accessibility, Silk Test Workbench cannot fully recognize a `DirectUIHwnd` control, because there is no public information about this control. Internet Explorer uses two `DirectUIHwnd` controls, one of which is a popup at the bottom of the browser window. This popup usually shows the following:

- The dialog box asking if you want to make Internet Explorer your default browser.
- The download options **Open**, **Save**, and **Cancel**.

When you start a project in Silk Test Workbench and record locators against the `DirectUIHwnd` popup, with accessibility disabled, you will see only a single control. If you enable Accessibility you will get full recognition of the `DirectUIHwnd` control.

Using Accessibility

Win32 uses the Accessibility support for controls that are recognized as generic controls. When Win32 locates a control, it tries to get the accessible object along with all accessible children of the control.

Objects returned by Accessibility are either of the class `AccessibleControl`, `Button` or `CheckBox`. `Button` and `Checkbox` are treated specifically because they support the normal set of methods and properties defined for those classes. For all generic objects returned by Accessibility the class is `AccessibleControl`.

Example

If an application has the following control hierarchy before Accessibility is enabled:

- Control
  - Control
- Button

When Accessibility is enabled, the hierarchy changes to the following:

- Control
  - Control
    - Accessible Control
    - Accessible Control
      - Button
- Button

Enabling Accessibility

If you are testing a Win32 application and cannot recognize objects, you should first enable Accessibility. Accessibility is designed to enhance object recognition at the class level.

To enable Accessibility:

1. Click **Tools** > **Options**. The **Options** dialog box opens.
2. Click **Advanced**.
3. Select the **Use Microsoft Accessibility** option. Accessibility is turned on.

Overview of Silk Test Workbench Support of Unicode Content

The Open Agent is Unicode-enabled, which means that the Open Agent is able to recognize double-byte (wide) languages.

With Silk Test Workbench you can test applications that contain content in double-byte languages such as Chinese, Korean, or Japanese (Kanji) characters, or any combination of these.

The Open Agent supports the following:

- Localized versions of Windows.
- International keyboards and native language Input Method Editors (IME).
- Passing international strings as parameters to test cases, methods, and so on, and comparing strings.
- Reading and writing text files in multiple formats: ANSI, Unicode, and UTF-8.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Before testing double-byte characters with Silk Test Workbench

Testing an internationalized application, particularly one that contains double-byte characters, is more complicated than testing an application that contains strictly English single-byte characters. Testing an internationalized application requires that you understand a variety of issues, from operating system support, to language packs, to fonts, to working with IMEs and complex languages.

Before you begin testing your application using Silk Test Workbench, you must do the following:

- Meet the needs of your application under test (AUT) for any necessary localized OS, regional settings, and required language packs.
- Install the fonts necessary to display your AUT.
- If you are testing an application that requires an IME for data input, install the appropriate IME.

Text Recognition Support

Text recognition methods, together with image recognition methods, enable you to conveniently interact with test applications that contain highly customized controls, which cannot be identified using object recognition. You can use *text clicks* instead of coordinate-based clicks to click on a specified text string within a control.

For example, you can simulate selecting the first cell in the second row of the following table:

CustomerName	FirstOrder	ID	IsActive	CreditCard
Bob Villa	01.01.2008	0	☑	MasterCard
Brian Miller	02.01.2008	1	☐	Visa
Caral Rudd	03.01.2008	2	☑	American Ex...
Dan Rundgren	04.01.2008	3	☐	MasterCard
Devie Yingstein	05.01.2008	4	☑	Visa

Specifying the text of the cell results in the following code:

```
'VB code
table.TextClick("Brian Miller")
```

Text recognition methods are supported for the following technology domains:

- Win32.
- WPF.
- Windows Forms.
- Java SWT and Eclipse.
- Java AWT/Swing.

Note: For Java Applets, and for Swing applications with Java versions prior to version 1.6.10, text recognition is supported out-of-the-box. For Swing applications with Java version 1.6.10 or later, which do not support Direct3D, you have to add the following command-line element when starting the application:

```
-Dsun.java2d.d3d=false
```

For example:

```
javaw.exe -Dsun.java2d.d3d=false -jar mySwingApplication.jar
```

Text recognition is not supported for Java Applets and Swing applications that support Direct3D.

- Internet Explorer.

Note: Text recognition does not work with controls that are not visible on the screen. For example, you cannot use text recognition for a text that is scrolled out of view.

Note: Text recognition might not work if the font that is used in the target text is not installed on the machine on which the test is executed.

Text recognition methods

The following methods enable you to interact with the text of a control:

`TextCapture`	Returns the text that is within a control. Also returns text from child controls.
`TextClick`	Clicks on a specified text within a control. Waits until the text is found or the *Object resolve timeout*, which you can define in the synchronization options, is over.
`TextRectangle`	Returns the rectangle of a certain text within a control or a region of a control.
`TextExists`	Determines whether a given text exists within a control or a region of a control.

Text click recording

Text click recording is enabled by default. To disable text click recording, click **Tools** > **Options** > **Record** > **General** and set the value of **Record 'TextClick'** to `No`.

When text click recording is enabled, Silk Test Workbench records `TextClick` methods instead of clicks with relative coordinates. Use this approach for controls where `TextClick` recording produces better results than normal coordinate-based clicks. You can insert text clicks in your script for any control, even if the text clicks are not recorded.

If you do not whish to record a `TextClick` action, you can turn off text click recording and record normal clicks.

The text recognition methods prefer whole word matches over partially matched words. Silk Test Workbench recognizes occurrences of whole words previously than partially matched words, even if the partially matched words are displayed before the whole word matches on the screen. If there is no whole word found, the partly matched words will be used in the order in which they are displayed on the screen.

Example

The user interface displays the text *the hostname is the name of the host*. The following code clicks on *host* instead of *hostname*, although *hostname* is displayed before *host* on the screen:

```
'VB code
control.TextClick("host")
```

The following code clicks on the substring *host* in the word *hostname* by specifying the second occurrence:

```
'VB code
control.TextClick("host", 2)
```

Custom Controls

Silk Test Workbench provides the following features to support you when you are working with custom controls:

- The *dynamic invoke* functionality of Silk Test Workbench enables you to directly call methods, retrieve properties, or set properties on an actual instance of a control in the application under test (AUT).

- The *class mapping* functionality enables you to map the name of a custom control class to the name of a standard Silk Test class. You can then use the functionality that is supported for the standard Silk Test class in your test.

  Silk Test Workbench supports managing custom controls over the UI for the following technology domains:

  - Win32
  - Windows Presentation Foundation (WPF)
  - Windows Forms
  - Java AWT/Swing
  - Java SWT
- You can add code to the AUT to test custom controls.
- The **Manage Custom Controls** dialog box enables you to specify a name for a custom control that can be used in a locator and also enables you to write reusable code for the interaction with the custom control.

Note: For custom controls, you can only record methods like `Click`,`TextClick`, and `TypeKeys` with Silk Test Workbench. You cannot record custom methods for custom controls except when you are testing Apache Flex applications.

Dynamic Invoke

Dynamic invoke enables you to directly call methods, retrieve properties, or set properties, on an actual instance of a control in the application under test. You can also call methods and properties that are not available in the Silk Test Workbench API for this control. Dynamic invoke is especially useful when you are working with custom controls, where the required functionality for interacting with the control is not exposed through the Silk Test Workbench API.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

Call dynamic methods on objects with the `Invoke` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Call multiple dynamic methods on objects with the `InvokeMethods` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.

For example, to call a method named `SetTitle`, which requires the title to be set as an input parameter of type string, on an actual instance of a control in the application under test, type the following:

```
control.Invoke("SetTitle", "my new title")
```

Note: Typically, most properties are read-only and cannot be set.

Note: Reflection is used in most technology domains to call methods and retrieve properties.

Note: You cannot dynamically invoke methods for DOM elements.

Frequently Asked Questions About Dynamic Invoke

This section includes a collection of questions that you might encounter when you are dynamically invoking methods to test custom controls.

Which Methods Can I Call With the Invoke Method?

To get a list of all the methods that you can call with the `Invoke` method for a specific test object, you can use the `GetDynamicMethodList`. To view the list, you can for example print it to the console or view it in the debugger.

Why Does an Invoke Call Return a Simple String when the Expected Return is a Complex Object?

The `Invoke` method can only return simple data types. Complex types are returned as string. Silk Test Workbench uses the `ToString` method to retrieve the string representation of the return value. To call the individual methods and read properties of the complex object that is returned by the first method invocation, use `InvokeMethods` instead of `Invoke`.

How Can I Simplify My Scripts When I Use Many Calls To InvokeMethods?

When you extensively use `InvokeMethods` in your scripts, the scripts might become complex because you have to pass all method names as strings and all parameters as lists. To simplify such complex scripts, create a static method that interacts with the actual control in the AUT instead of interacting with the control through `InvokeMethods`. For additional information, see *Adding Code to the Application Under Test to Test Custom Controls*.

Adding Code to the Application Under Test to Test Custom Controls

When you are testing Windows Forms applications or WPF applications, and you want to test complex custom controls or custom controls that you cannot test by simply using the `Invoke` and `InvokeMethods` methods, you can create a static method that interacts with the actual control in the application under test (AUT) and you can add this code to the AUT.

The benefit for you from adding code to the AUT is that the code in the AUT can use regular method calls for interacting with the control, instead of using the reflection-like style of calling methods with the dynamic invoke methods. Therefore you can use code completion and IntelliSense when you are writing you code. You can then call the code in the AUT with a simple invoke call, where you pass the control of interest as a parameter.

You can add code to the AUT in the following ways:

- Compile the code into the AUT. The implementation is simple, but you will be changing the AUT, which you might not want to do.
- Inject code to the AUT at runtime by using the `LoadAssembly` method in a test script. This requires more effort than compiling the code into the AUT, but the injected code will be located close to the test code. The `LoadAssembly` method is available for the classes `WPFWindow` and `FormsWindow`.

Example: Testing the UltraGrid Infragistics control

This example demonstrates how you can retrieve the content of an `UltraGrid` control. The `UltraGrid` control is included in the `NETAdvantage for Windows Forms` library which is provided by Infragistics. You can download a trial of the library from *http://www.infragistics.com/products/windows-forms/downloads*.

To create the `UltraGridUtil` class, perform the following actions:

1. Create a new test script. Call the new script `AUTExtensions`.

Note: Your AUT must use .NET version 4 or later. When you compile the .NET script a .NET 4.0 assembly is created and this assembly can only be loaded into an AUT that uses the same or a later .NET version. If your AUT uses a .NET

version prior to 4.0, use Visual Studio to create an class library project with the `UltraGridUtil` class.

2. Add references to the required dependencies to the script. For example, for Infragistics version 12.2 you need to reference the following assemblies:

   - Infragistics4.Shared.v12.2
   - Infragistics4.Win.UltraWinGrid.v12.2
   - Infragistics4.Win.v12.2

   If you are not sure which version of Infragistics is used in your AUT you can use the **Process Explorer** tool from Microsoft to see which assemblies are loaded in your AUT.

   a. Replace the content of the `AUTExtensions` script with the following code:

```
' VB code
Public Class UltraGridUtil

  Public Shared Function GetContents(ultraGrid As
Infragistics.Win.UltraWinGrid.UltraGrid) As List(Of List(Of
String))
    Dim contents = New List(Of List(Of String))
    For Each row In ultraGrid.Rows
      Dim rowContents = New List(Of String)
      For Each cell In row.Cells
        rowContents.Add(cell.Text)
      Next
      contents.Add(rowContents)
    Next
    Return contents
  End Function

End Class
```

Note: The `Shared` modifier makes the `GetContents` method a static method.

3. Load the assembly into the AUT during playback.

   - Open an existing test script or create a new test script.
   - Add a reference to the `AUTExtensions` script.
   - Add the following code to your test script:

```
' VB code
mainWindow.LoadAssembly(GetType(UltraGridUtil).Assembly.Loca
tion)
```

Note: Silk Test Workbench compiles every script into a separate .NET assembly.

4. Call the static method of the injected code in order to get the contents of the `UltraGrid`:

```
'VB code
Dim ultraGrid = mainWindow.Control("@automationId='my grid'")
Dim contents As IList =
mainWindow.Invoke("UltraGridUtil.GetContents", ultraGrid)
```

Frequently Asked Questions About Adding Code to the AUT

This section includes a collection of questions that you might encounter when you are adding code to the AUT to test custom controls.

Why is Code That I Have Injected Into the AUT With the LoadAssembly Method Not Updated in the AUT?

If code in the AUT is not replaced by code that you have injected with the `LoadAssembly` method into the AUT, the assembly might already be loaded in your AUT. Assemblies cannot be unloaded, so you have to close and re-start your AUT.

Why Do the Input Argument Types Not Match When I Invoke a Method?

If you invoke a method and you get an error that says that the input argument types do not match, the method that you want to invoke was found but the arguments are not correct. Make sure that you use the correct data types in your script.

If you use the `LoadAssembly` method in your script to load an assembly into the AUT, another reason for this error might be that your assembly is built against a different version of the third-party library than the version that is used by the AUT. To fix this problem, change the referenced assembly in your script. If you are not sure which version of the third-party library is used in your AUT, you can use the **Process Explorer** tool from Microsoft.

How Do I Fix the Compile Error when an Assembly Can Not Be Copied?

When you have tried to add code to the AUT with the `LoadAssembly` method, you might get the following compile error:
Could not copy '<assembly_name>.dll' to '<assembly_name>.dll'. The process cannot access the file.
The reason for this compile error is that the assembly is already loaded in the AUT and cannot be overwritten.

To fix this compile error, close the AUT and compile your script again. To avoid that problem in the first place, copy the assembly into your script, for example to a temporary directory, and then inject the copied assembly with the `LoadAssembly` method. For example:

```
' VB code
Dim typeThatContainsYourStaticMethod = GetType (UltraGridUtil)
Dim assemblyToLoadIntoAUT = typeThatContainsYourStaticMethod.Assembly.Location
Dim tempFile = IO.Path.GetTempFileName()
IO.File.Copy(assemblyToLoadIntoAUT, tempFile, True )
mainWindow.LoadAssembly(tempFile)
```

Testing Apache Flex Custom Controls

Silk Test Workbench supports testing Apache Flex custom controls. However, by default, Silk Test Workbench cannot record and playback the individual sub-controls of the custom control.

For testing custom controls, the following options exist:

- Basic support

  With basic support, you use dynamic invoke to interact with the custom control during replay. Use this low-effort approach when you want to access properties and methods of the custom control in the test application that Silk Test Workbench does not expose. The developer of the custom control can also add methods and properties to the custom control specifically for making the control easier to test. A user can then call those methods or properties using the dynamic invoke feature.

  The advantages of basic support include:

  - Dynamic invoke requires no code changes in the test application.
  - Using dynamic invoke is sufficient for most testing needs.

  The disadvantages of basic support include:

  - No specific class name is included in the locator, for example Silk Test Workbench records `//FlexBox` rather than `//FlexSpinner`.

- Only limited recording support.
- Silk Test Workbench cannot replay events.

For more details about dynamic invoke, including an example, see *Dynamically Invoking Apache Flex Methods*.

- Advanced support

  With advanced support, you create specific automation support for the custom control. This additional automation support provides recording support and more powerful play-back support. The advantages of advanced support include:

  - High-level recording and playback support, including the recording and replaying of events.
  - Silk Test Workbench treats the custom control exactly the same as any other built-in Apache Flex control.
  - Seamless integration into Silk Test Workbench API
  - Silk Test Workbench uses the specific class name in the locator, for example Silk Test Workbench records `//FlexSpinner`.

  The disadvantages of advanced support include:

  - Implementation effort is required. The test application must be modified and the Open Agent must be extended.

Managing Custom Controls

You can create custom classes for custom controls for which Silk Test Workbench does not offer any dedicated support. Creating custom classes offers the following advantages:

- Better locators for scripts and visual tests.
- An easy way to write reusable code for the interaction with the custom control.

Example: Testing the UltraGrid Infragistics control

Suppose that a custom grid control is recognized by Silk Test Workbench as the generic class `Control`. Using the custom control support of Silk Test Workbench has the following advantages:

Better object recognition because the custom control class name can be used in a locator.	Many objects might be recognized as `Control`. The locator requires an index to identify the specific object. For example, the object might be identified by the locator `//Control[13]`. When you create a custom class for this control, for example the class `UltraGrid`, you can use the locator `//UltraGrid`. By creating the custom class, you do not require the high index, which would be a fragile object identifier if the application under test changed.
You can implement reusable playback actions for the control in scripts.	When you are using custom classes, you can encapsulate the behavior for getting the contents of a grid into a method by adding the following code to your custom class, which is the class that gets generated when you specify the custom control in the user interface. Typically, you can implement the methods in a custom control class in one of the following ways:

- You can use methods like `Click`, `TypeKeys`, `TextClick`, and `TextCapture`.
- You can dynamically invoke methods on the object in the AUT.
- You can dynamically invoke methods that you have added to the AUT. This is the approach that is described in this example.

You can use the following code to call the static method that is defined in the example in *Adding Code to the Application Under Test to Test Custom Controls*. The method `GetContents` is added into the generated class `UltraGrid`.

```
' VB code
Partial Public Class UltraGrid

  Public Function GetContents() As
IList
    Return
Invoke("UltraGridUtil.GetContents",
Me)
  End Function

End Class
```

When you define a class as a custom control, you can use the class in the same way in which you can use any built-in class, for example the `Dialog` class.

```
' VB code
Dim ultraGrid As UltraGrid =
mainWindow.UltraGrid("@automationId=
'my grid'")
Dim contents =
ultraGrid.GetContents()
```

Supporting a Custom Control

Silk Test Workbench supports managing custom controls over the UI for the following technology domains:

- Win32
- Windows Presentation Foundation (WPF)
- Windows Forms
- Java AWT/Swing
- Java SWT

To create a custom class for a custom control for which Silk Test Workbench does not offer any dedicated support.

1. Click **Tools** > **Manage Custom Controls**. The **Manage Custom Controls** dialog box opens.
2. In the **.NET Script for Custom Controls** field, type in a name or click **Browse** to select the script that will contain the custom control.

   The .NET script for custom controls is saved in the Common project. For all new databases, the default .NET script is called *CustomControlScript*. This script is automatically created for the user during the initial login to the database.

Note: The .NET script is required. If no script is selected in the **.NET Script for Custom Controls** field, the **OK** button is disabled. Select a script to enable the button. For existing databases, if you have write access to the Common project and if there is no .NET script for custom controls set, Silk Test Workbench tries to save a new script in the Common project. If a script with the name *CustomControlScript* already exists, and you have write access to the Common project, Silk Test Workbench tries to save the script with a number appended to the end of the script name, by sequentially trying the names *CustomControlScript2*, *CustomControlScript3*, and so on.

3. Click on the tab of the technology domain for which you want to create a new custom class.
4. To automatically select the control, click **Identify** and the Silk Test Workbench switches to identify mode and you can select a control as usual. The **Custom control class name** and **Silk Test base class** fields are filled out in the current tab based on the identified control. Otherwise, continue with the **Add** steps below.
5. Click **Add**. A new row is added to the list of custom controls.
6. If you have chosen to manually add a custom control to the list:
   a) In the **Silk Test base class** column, select an existing base class from which your class will derive.
      This class should be the closest match to your type of custom control.
   b) In the **Custom control class name** column, enter the fully qualified class name of the class that is being mapped.
      For example: `Infragistics.Win.UltraWinGrid.UltraGrid`. You can use the wildcards *?* and * in the class name.
   c) If you are working in **Win32**, click the check box in the next column if you want to change the way you refer to the class. If you turn off the check box, the name will default to the same as the **Silk Test base class**. The **Silk Test class** column is now enabled.
   d) In the **Silk Test class** column, click **F2** and enter the name to use to refer to the class.
      This is what will be seen in locators. For example: `//UltraGrid` instead of `//Control[13]`.

      Note: After you add a valid class, it will become available in the **Silk Test base class** list. You can then reuse it as a base class.
7. *Only for Win32 applications:* In the **Use class declaration** column, set the value to **False** to simply map the name of a custom control class to the name of a standard Silk Test class.
   When you map the custom control class to the standard Silk Test class, you can use the functionality supported for the standard Silk Test class in your test. Set the value to **True** to additionally use the class declaration of the custom control class.
8. Click **OK**.
9. *Only for scripts:*
   a) Add custom methods and properties to your class for the custom control.
   b) Use the custom methods and properties of your new class in your script.

      Note: The custom methods and properties are not recorded.

Note: Do not rename the custom class or the base class in the script file. Changing the generated classes in the script might result in unexpected behavior. Use the script only to add properties and methods to your custom classes. Use the **Manage Custom Controls** dialog box to make any other changes to the custom classes.

Custom Controls Options

Tools > **Manage Custom Controls**.

Silk Test Workbench supports managing custom controls over the UI for the following technology domains:

- Win32
- Windows Presentation Foundation (WPF)

- Windows Forms
- Java AWT/Swing
- Java SWT

In the **.NET Script for Custom Controls**, define the script file into which the new custom classes should be generated.

The .NET script for custom controls is saved in the Common project. For all new databases, the default .NET script is called *CustomControlScript*. This script is automatically created for the user during the initial login to the database.

Note: The .NET script is required. If no script is selected in the **.NET Script for Custom Controls** field, the **OK** button is disabled. Select a script to enable the button. For existing databases, if you have write access to the Common project and if there is no .NET script for custom controls set, Silk Test Workbench tries to save a new script in the Common project. If a script with the name *CustomControlScript* already exists, and you have write access to the Common project, Silk Test Workbench tries to save the script with a number appended to the end of the script name, by sequentially trying the names *CustomControlScript2*, *CustomControlScript3*, and so on.

When you map a custom control class to a standard Silk Test class, you can use the functionality supported for the standard Silk Test class in your test. The following **Custom Controls** options are available:

Option	Description
Silk Test base class	Select an existing base class to use that your class will derive from. This class should be the closest match to your type of custom control.
Silk Test class	Enter the name to use to refer to the class. This is what will be seen in locators.
Custom control class name	Enter the fully qualified class name of the class that is being mapped. You can use the wildcards *?* and * in the class name.
Use class declaration	This option is available only for Win32 applications. By default `False`, which means the name of the custom control class is mapped to the name of the standard Silk Test class. Set this setting to `True` to additionally use the class declaration of the custom control class.

Note: After you add a valid class, it will become available in the **Silk Test base class** list. You can then reuse it as a base class.

Example: Setting the options for the UltraGrid Infragistics control

To support the `UltraGrid` Infragistics control, use the following values:

Option	Value
Silk Test base class	`Control`
Silk Test class	`UltraGrid`
Custom control class name	`Infragistics.Win.UltraWinGrid.UltraGrid`

Measuring Execution Time

You can use methods and properties provided by the `Timer` class to measure the time that your tests require to execute. For additional information, see *Timer Class*.

Among other usages, these methods and properties are used for the timing of test executions that are triggered from Silk Performer. For additional information on integrating Silk Test Workbench with Silk Performer, refer to the *Silk Performer Help*.

Slowing Down Tests

Some applications under test might require extensive loading of application data in the UI, and might not be finished on time with loading objects that are required for replaying a test. To successfully replay tests on such an AUT, you can check for the existence of an object before performing an action on it, or you can add sleeps before performing an action.

Note: Micro Focus does not recommend generally adding sleeps to tests, because in most cases Silk Test Workbench will automatically detect if an object is available, and sleeps might severely reduce the performance of tests.

1. To check if an object is available in the AUT, use the Exists method.

   For example, to wait for six seconds for the button INPUT to become available, add the following line to your test script:

   ```
   'VB .NET code
   browserWindow.Exists("//INPUT", 6000)
   ```

2. To add a sleep before performing an action on a control, use the `Sleep` method.

   For example, to sleep for six seconds, add the following line to your test script:

   ```
   'VB .NET code
   System.Threading.Thread.Sleep(6000)
   ```

Debugging Tests

Describes the process for debugging visual tests and scripts.

Debugging Visual Tests

Errors encountered during playback can be caused by a variety of factors, such as changes in the test application, improper visual test step flow, or environmental changes. Quickly diagnosing and fixing these errors using debugging features minimizes visual test maintenance and allows a more efficient team testing effort.

Visual test debugging enables you to temporarily suspend playback to manage, examine, reset, or step through playback. This lets you identify exactly where an error may be occurring. You can edit steps during debugging, so you can often resolve an error in a step before re-executing the step.

Suspending Visual Test Playback at Selected Points

You can configure visual tests to suspend at set points during playback and enter debug mode. The points at which you set a visual test to suspend on playback are called *breakpoints*. Breakpoints let you manage playback to facilitate debugging, and can help isolate where a visual test fails. Breakpoints help to analyze how a visual test plays back even if there are no errors in the visual test. When you set a breakpoint, playback suspends before executing the step where the breakpoint is set and enters debug mode.

Setting breakpoints allows a visual test to play back to any specific point of interest. Place breakpoints at specific steps where you want to suspend playback, such as any point where you want to see the state of data contained in variables in the Local Variables window. Playback executes to the first breakpoint and suspends until you tell the visual test to proceed.

1. Open the visual test in which you want to set a breakpoint.
2. Select the step where playback is to be suspended.
3. Choose **Debug** > **Set/Clear Breakpoint**. A breakpoint icon appears in the column next to the step number. Playback executes to the set breakpoint. When playback suspends at a breakpoint, the visual test highlights the breakpoint step in yellow with a yellow arrow pointing to the step.

The visual test displays in debug mode where you can step through the playback execution, control the step playback execution, or playback to/from certain points in the visual test.

Tip: Select a step with a breakpoint and press **F9** to delete the breakpoint from the selected step. Press **Ctrl+Shift+F9** to delete all breakpoints from all open visual tests and VB .NET scripts.

Stepping Through Visual Test Playback from a Selected Point

During debugging, you can execute the visual test line-by-line from where playback was suspended by a breakpoint.

1. Set a breakpoint at a specific step in the visual test.
2. Press F5 to playback the visual test. Playback executes to the breakpoint, suspends and enters debug mode. The next step to be played back is highlighted in yellow in the **Test Steps** pane.
3. Press F8 to playback the highlighted step.

If the step executes successfully, playback re-enters Debug mode and the next step to be played back highlights in yellow. If the step does not playback successfully, the **Playback Error** dialog box opens.

4. Press F8 to step through playback of the remaining steps in the visual test. Playback executes to the set breakpoint. The visual test displays in debug mode where you can step through the playback execution, control the step playback execution, or playback to/from certain points in the visual test. While stepping through a visual test, use the Local Variables window to get a snapshot of variable use at critical points in the execution of any test. During debugging, you can also configure visual tests to playback from a specific step.

Controlling Step Execution During Visual Test Playback

While debugging a visual test or at a breakpoint, you can control which step sequentially executes next in the visual test. By moving the execution point, you can resume playback at a selected step in another part of the visual test without executing any other steps.

The **Set Run pointer/next Statement** command sets the execution point to the step that you choose. This feature is only available while at a breakpoint during debugging.

Use the **Set Run pointer/next Statement** command when you want to re-run a statement within the current procedure or to skip over steps in your visual test you do not want to play back. While debugging a visual test, this feature lets you control flow to aid in error recognition and reduce the time needed to diagnose and fix errors.

1. While debugging or at a breakpoint during playback, select the next step you want to execute.

   Select any step in the visual test.
2. Choose **Debug** > **Set Run pointer/next Statement**. The yellow execution arrow moves to the selected step and highlights the step in yellow.
3. Press F5 to resume playback at the selected step or press F8 to playback only the selected step. After the selected step executes, playback suspends, and Silk Test Workbench returns to debug mode. The execution point goes to the next sequential step.

Editing Visual Tests During Debugging

You can edit and save visual tests during debugging. When you save changes, a new version of the visual test is created and becomes the active visual test.

While stepping through a visual test during debugging, you can make changes that take effect during playback. You can make the following types of edits in a visual test during debugging:

- Adding flags.
- Editing properties of an existing step.
- Copying and pasting test steps.
- Setting and clearing breakpoints.
- Editing of visual tests that are inserted into other visual tests. For example, if you insert *visualtest2* into *visualtest1*, you can edit *visualtest2* without any restrictions.

Stepping Through a Visual Test in Debug Mode

You can play back visual tests one step at a time while in debug mode. Known as *stepping*, this allows you to trace through steps during playback to see the order in which the test steps execute, any data returned by a specific step, and if a step executes successfully.

In debug mode, control playback execution by using the following commands:

Step Into (F8)	Executes playback one step at a time. **Step Into** is useful to trace through each step, and steps into other visual tests or scripts that are inserted into the visual test being played back. Each inserted visual test or script is also executed one step or line at a time. **Step Into** is useful for detailed analysis of a test, and lets you see the effect of each step on variable usage and test application interaction.
Step Over (`Shift + F8`)	Executes an entire visual test or script inserted into another visual test as if it were a single step. Use **Step Over** when playback is in debug mode for a step that plays back a visual test or script. This plays back the inserted visual test or script in its entirety. Once the inserted visual test or script plays back in its entirety, playback suspends in debug mode at the next step in the original visual test. Using **Step Over** at a step other than one that plays back another visual test or script has the same effect as using **Step Into**. Only the next step executes before playback suspends and re-enters debug mode.
Step Out (`Ctrl + Shift + F8`)	Executes all remaining steps in a visual test or script being played back from another visual test, then suspends playback at the next step in the original visual test. Use **Step Out** when playback is suspended at a step in a visual test or code line that has been inserted in another visual test, and you want to playback the remaining visual test or script and return to the original visual test. When playback executes the remainder of the inserted script or visual test, it suspends and re-enters debug mode at the next sequential line in the original visual test.
Run To Cursor (`Ctrl + F8`)	Allows you to select a step where you want playback suspended. This allows you to "step over" selected sections of a visual test or script. Use **Run To Cursor** to playback the visual test or script and stop playback at a point just before a run-time error occurs. This lets you stop playback at a specific line or statement without having to insert breakpoints. Once playback stops, you can continue using one of the other debug options.
Run From Cursor	Plays back the visual test from the currently selected test step.

Stepping commands are accessed from the **Debug** menu.

Handling Errors in Visual Tests to Ensure Playback

Application testing reveals errors in the application being tested. When testing software applications using Silk Test Workbench, these errors can cause visual tests to not play back to completion. For example, if a test step tries to enter text in a TextField that is not found during playback, an object not found error occurs and playback stops. Users can then enter debug mode to determine the cause of the error.

Error handling can eliminate the need for debugging playback errors by telling the visual test to detect errors, and how to respond to them when they occur. Error handling enables the visual test to playback to completion and provides insight into errors encountered during playback.

Setting Up a Visual Test to Automatically Respond to Errors Using the Properties Pane

You can configure visual tests to anticipate and automatically respond to playback errors. Set up visual tests to handle errors within its own steps, or to open another visual test containing steps that provide specific error handling instructions. To set up a visual test to handle errors, create an `On Error` step.

1. Open the visual test in which you want to set up error handling.
2. Select the step that precedes where you want to create the `On Error` step.
3. Choose **Insert** > **Error Handling**. The `On Error` step is created after the selected step. Properties for the `On Error` step display in the **Properties** pane.

4. Select the newly created step to display its properties in the **Properties** pane.
5. In the **Properties** pane for the `On Error` step, click the toolbar buttons to group properties by category or to sort alphabetically.
6. Update the **On error go to** properties to assign the playback setting and its value.

   If the **Action to take** property for the error handler is **Go to**, specify a label in the visual test for the error handler to navigate to.

The `On Error` step executes whenever an error is encountered during playback.

Setting Up a Visual Test to Automatically Respond to Errors Using the Test Logic Designer

You can configure visual tests to anticipate and automatically respond to playback errors. Set up visual tests to handle errors within its own steps, or to open another visual test containing steps that provide specific error handling instructions. To set up a visual test to handle errors, use the **Test Logic Designer** wizard.

1. Open the visual test in which you want to set up error handling.
2. Select the step that precedes where you want to create the error handling step.
3. Choose **Insert** > **Test Logic** > **Error Handling**. The **Test Logic Designer** wizard opens to the **Welcome** page.
4. Click **Next**. The **Select a Logic Type** page opens.
5. Click **The error during playback of the visual test (Playback Error)** and then click **Next**. The **Build the Error Handler** page opens.
6. From the **Whenever a playback error occurs take the following action** list, select an action from one of the following choices.

Show default playback error dialog	Displays the **Playback Error** dialog box when a playback error occurs, which is used to determine a playback action to take when an error occurs. Use the **Playback Error** dialog box to enter debug mode to diagnose errors in a visual test.
End playback of current visual test	Ends visual test playback when an error occurs. No other steps in the visual test play back. When playback ends, the **Playback Complete** dialog box appears, which determines the action to take when playback completes.
Retry playback of the step on which the error occurred	Plays back the visual test specified in the **Asset** property, then attempts to execute the step that caused the error. If no visual test is specified in the **Asset** property, then playback simply attempts to re-execute the step that caused the error. You can specify the number of replay attempts in the **Number of retries** property. The default number is 5. If the step execution is not successful when the specified number of retries is reached, a playback error is generated.
Resume playback at the next step	Plays back the visual test specified in the **Asset** property, then attempts to execute the step immediately after the step that caused the error. The step that caused the error is not re-executed.
Go to a label	Plays back the visual test specified in the **Asset** property, then navigates to the predefined Label step specified in the **Label** property and continues playback. Using **Go to** allows that visual test to skip steps that could potentially contain failures similar to the one that occurred during playback, and navigate to another section of the test.
First playback another visual test	Plays back a different visual test, which you specify, and then you specify what action happens next. To determine what occurs after the alternate test runs, choose to end the test, retry the step where the error occurred, go to a label, or resume the original test where the error occurred.

7. If you specify **Go to a label**, from the **Select a label to Go to** list box, select the label to which you want the error handler to navigate.
8. If you specify **First playback another visual test**, perform the following steps.
   a) Click **<Insert a Visual test>**, choose the asset that you want to playback, and then click **OK**.
   b) From the **And then** list box, select which action you want to occur next.

      If you specify **Go to a label**, from the **Select a label to Go to** list box, select the label to which you want the error handler to navigate.
9. Click **Next**. The **Summary** page shows the test step description for the new error handling step.
10. Click **Finish**.

Printing a Visual Test

Print visual tests to save the content of your tests.

1. Open the visual test that you want to print.
2. Choose **File** > **Print**. The **Print** dialog box opens.
3. Select the printer from the list.
4. *Optional:* Click **Properties** to change the page setup settings, for example margins and orientation.
5. *Optional:* In the **Page Range** section, specify whether to print the entire document or only selected pages.
6. *Optional:* In the **Number of copies** filed, specify the number of copies to print.
7. *Optional:* Check the **Collate** check box to print the corresponding page of every copy together.

   This check box is only enabled when you have selected to print more than one copy.
8. Specify which steps should be printed.
   - Click **Current view** to print all steps as they display in the **Test Steps** pane.
   - Click **Selection** to print only the steps that are selected in the **Test Steps** pane.
9. To customize the print output, click **Options** and select the type of information that you want to include in the printed output.

   Options include:

   Include — Specifies the type of content to include.
   - **Visual test summary** – Includes a summary of the visual test that contains the name, project, creator, creation date, version, and description of the visual test. This summary appears on the first page of the printed output.
   - **Latest result summary** – Includes the summary page of the latest result of the visual test. This summary appears on the last page of the printed output.
   - **Group screens** – Includes the group screen snapshots, which contain the main context of the current dialog box or Web page, in the printed output. The test steps that follow the group screen correspond with a single action that occurs on the context of the current dialog box or Web page. The group screen corresponds with the `Using` step in the **Test Steps** pane. The group screen snapshots appear in the printed output as they do in the **Screen Preview** pane.
   - **Disabled steps** – Includes any disabled steps, which appear in gray text in the printed output.

   Include for each step — Specifies the content to include for each step.
   - **Screen** – Includes any associated screen snapshots for each individual step. Individual screens correspond with a `(step screen)`, which is associated with a specific action on the current dialog box or Web page. Step screen snapshots appear in the printed output as they do in the **Screen Preview** pane.

- **Description** – Includes the step description for each step. The step description in the printed output appears as it does in the **Step description** property of the **Properties** pane.
- **Flag details** – Includes flag details such as the flag description, creator, modifier, modification date, and assigned user.

Include the following properties
Specifies the properties to include for each step.
- **All properties** – Includes all properties for each step.

Default
Click this button to select the default print options.

10. *Optional*: Click **Preview** to display how the result information will look when printed. The **Preview** window opens.
11. Click **OK**.

Debugging Scripts

Run-time errors encountered during playback can be caused by a variety of factors, such as changes in the test application, script coding mistakes, or environmental changes. Quickly diagnosing and fixing these errors by debugging scripts minimizes script maintenance efforts and facilitates a more efficient team testing effort.

Script debugging lets you temporary suspend script playback in the development environment to manage, examine, reset, step through, or continue script playback.

Silk Test Workbench also provides an **Output** window where you can review automation code and append additional user-defined output. For instance, you can use the `Console.Write` and the `ConsoleWriteLine` commands to output additional information.

In addition, you can use the Local Variables window to drill down and see the values of all variables in the script.

Script Reliability

Application testing reveals errors in the application being tested. When testing software applications, these errors can cause scripts to not execute to completion.

The ability of scripts to reliably play back to completion is essential to successful testing. Synchronization issues, and expected and unexpected application errors may cause scripts to fail before playing back to completion. Silk Test Workbench provides built-in option settings and scripting features that allow scripts to adjust to synchronization and application issues and play back reliably in any testing environment.

When playing back scripts, some errors are the result of slow application response time, which causes the application to lose synchronization with the scripts that test it. These errors can be handled by adjusting Silk Test Workbench option settings that optimize synchronization between the scripts and the application being tested.

Stepping Through a Script in Debug Mode

You can play back scripts to execute one line at a time while in debug mode. Known as *stepping*, this allows a tester to trace through code during playback to see the order in which statements execute, which aids in debugging.

Stepping enables you to walk through a script in debug mode to closely examine its execution.

In debug mode, control playback execution by using the following commands:

Step Into (F8)	Executes the script one statement at a time. **Step Into** is useful to trace through each line in the code and steps into functions or embedded scripts. Each function or embedded script is also executed one code line at a time. **Step Into** is useful for detailed analysis of a test, and lets you see the effect of each statement on variable usage and test application interaction.  **Note:** You cannot step into an *iterator* while debugging a VB .NET script. The iterator is executed but the debugger will not show that the iterator was entered during execution. You can set a breakpoint inside the iterator, and the execution will stop at the breakpoint, but any parameters or local variables of the iterator will not be shown in the **Locals** window. For additional information on iterators, see *Iterators*.
Step Over (`Shift + F8`)	Executes each procedure as if it were a single statement. Use this instead of Step Into to go directly into a function, without stepping through the individual lines that make up the function.
Step Out (`Ctrl + Shift + F8`)	Executes all remaining code in a procedure as if it were a single statement, and exits to the next statement in the procedure that caused the procedure to be initially called.
Run To Cursor (`Ctrl + F8`)	Allows you to select a step where you want playback suspended. This allows you to "step over" selected sections of a visual test or script. Use **Run To Cursor** to playback the visual test or script and stop playback at a point just before a run-time error occurs. This lets you stop playback at a specific line or statement without having to insert breakpoints. Once playback stops, you can continue using one of the other debug options.
Set Run Pointer/ Next Statement (Ctrl + F9)	Continues execution with the statement at which the pointer is currently located, without executing any intermediate statements.

Stepping commands are accessed from the **Debug** menu.

Stepping Through Script Playback from a Selected Point

During debugging, you can execute the script from where playback was suspended by a breakpoint.

1. Set a breakpoint at a specific line in the script.
2. Press F5 to playback the script. Playback executes to the breakpoint, suspends and enters debug mode.
3. Press F8 to playback the script to the next breakpoint.

   If the code executes successfully, playback re-enters debug mode and the next code to be played back is highlighted in yellow. If the code does not playback successfully, the **Playback Error** dialog box opens.
4. Press F8 to step through playback of the remaining code in the script. Playback executes to the set breakpoint. The script displays in debug mode where you can step through the playback execution, control the step playback execution, or playback to/from certain points in the script. While stepping through a script, use the **Monitor Variables** window to get a snapshot of variable use at critical points in the execution of any test. During debugging, you can also configure scripts to playback from a specific line of the code.

Controlling Line Execution During Script Debugging

While debugging a script, you can control flow to aid in error recognition and to reduce the time needed to diagnose and fix errors.

The **Set Run pointer/next Statement** command sets the execution point to the line in your script that you choose. This feature is only available while at a breakpoint during debugging.

Use the **Set Run pointer/next Statement** command when you want to re-run a statement within the current procedure or to skip over statements in your script you do not want to play back. While debugging a script, this feature lets you control flow to aid in error recognition and reduce the time needed to diagnose and fix errors.

1. While debugging or at a breakpoint during playback, perform one of the following steps:
   - Drag the yellow arrow to the code line to be played back next.
   - Click to position the cursor on the next line to be played back and then choose **Debug** > **Set Run pointer/next Statement**.

   Note: In scripts, you can only set the next statement to a line in the same procedure as the current statement.
2. Press F5 to resume playback at the selected line or press F8 to playback only the selected line. After the selected line executes, playback suspends, and Silk Test Workbench returns to debug mode. The execution point goes to the next sequential line.

Example Script - Error Handling

The following script illustrates how you can construct a script to include error handling.

To include basic error handling in a script, include a `Catch ex As Exception` statement in your script followed by the object and the parameters that you want to test for an exception. For example, to test the Notepad application font type, you might type the following code:

```
Public Module Main
    Dim _desktop As Desktop = Agent.Desktop

    Public Sub Main()
        With _desktop.Window("@caption='Untitled - Notepad'")
            .TextField().TypeKeys("test", 0, False)
            .MenuItem("@caption='Font'").Select()
            Try
                With .Dialog("@caption='Font1'") ' throws an exception
because such a dialog does not exist
                    .ComboBox("@caption='Font style:'").Select("Regular")
                    .ComboBox("@caption='Size:'").Select("14")
                    .PushButton("@caption='Cancel'").Select()
                End With
            Catch ex As Exception
                With .Dialog("@caption='Font'")
                    .ComboBox("@caption='Font style:'").Select("Regular")
                    .ComboBox("@caption='Size:'").Select("14")
                    .PushButton("@caption='Cancel'").Select()
                End With
            End Try
        End With
    End Sub

End Module
```

Note: This example is meant to illustrate how you can implement basic error handling. You must add an application configuration and make any necessary adjustments for your test application to

successfully implement this code. If you cut and paste this example into your script, an error occurs because the application configuration is missing.

Inserting a Comment in a Script

Use the **Output** window to review automation code and append additional user-defined output to provide supplemental information about a script. For instance, you can add comments to provide supplemental information about a script. After playback of a script, the comment appears in the **Output** window.

1. In a script, you can add comments by using the `Console.Write` and the `Console.WriteLine` commands.

   The `Comment` method takes a string as a parameter. The contents of this string are saved in the **Output** window after playback of the script.
2. Play back the script. From the **Output** window, you can view the automation code and the comments that you added.

Stopping Script Playback at Selected Points

You can configure scripts to suspend at set points during playback and enter debug mode. The points at which you set a script to stop on playback are called *breakpoints*. Breakpoints let you manage playback to facilitate debugging, and can help isolate where a script fails. Breakpoints help to analyze how a script plays back even if there are no errors in the script. When you set a breakpoint, playback stops at the breakpoint and enters debug mode.

Setting breakpoints enables a script to play back to any specific point of interest. Place breakpoints at specific lines where you want to stop playback, such as any point where you want to see the state of data contained in variables in the **Monitor Variables** window. Playback executes to the first breakpoint and suspends until you tell the script to proceed.

1. Open the script in which you want to set a breakpoint.
2. Click the line where playback is to be suspended.
3. Click **Debug** > **Set/Clear Breakpoint**. A breakpoint icon appears in the column next to the line number. Playback executes to the set breakpoint. When playback suspends at a breakpoint, the script highlights the breakpoint line in yellow with a yellow arrow pointing to the line.

During debugging, you can control line execution during playback. You can also configure scripts to playback to or from a specific line.

Tip: Select a line with a breakpoint and press `F9` to delete the breakpoint from the selected line. Press `Ctrl+Shift+F9` to delete all breakpoints from a script.

Printing a Script

You can print the code that displays in the **Code** window for easy review and troubleshooting.

1. Open the script that you want to print.
2. Choose **File** > **Print**. The **Print** dialog box opens.
3. Select the printer from the list or click **Find Printer** to search for an additional printer.
4. *Optional:* Check the **Print to file** check box to print to a file, such as a PDF.
5. *Optional:* Click **Preferences** to change the page setup settings, for example margins and orientation.
6. *Optional:* In the **Page Range** section, specify whether to print the entire document or only selected pages.

   Click one of the following options.

   - **All** – Prints the entire document.

- **Selection** – Prints the selected code.
- **Pages** – Prints the specified pages.
- **Current Page** – Prints the current page.

7. *Optional:* In the **Number of copies** filed, specify the number of copies to print.
8. *Optional:* Check the **Collate** check box to print the corresponding page of every copy together. This check box is only enabled when you have selected to print more than one copy.
9. Click **Print**.

Playing Back Tests and Analyzing Results

Describes the process for playing back tests and analyzing the results of the playback.

Playing Back Visual Tests

Once a visual test is recorded, you can play it back at any time. Test steps execute in sequence from top to bottom. During playback, the visual test drives the test application according to the recorded or added actions. You can observe these actions taking place on the screen. The actions are performed at the fastest speed allowed by the application's responses. Visual tests can be played back at speeds much greater than manual execution.

Because visual test are recorded on the object level, without reference to an object's position on the screen, playback is also object-oriented. This means that Silk Test Workbench interacts directly with objects. It does not matter if there have been environmental changes or changes in the test application, visual tests will still play back properly.

Whenever a visual test is played back, Silk Test Workbench repeats all the actions recorded in the test and conducts the verifications or other actions that have been recorded or added.

Playing Back a Visual Test

When you play back a visual test, it performs each of the actions that you recorded or added.

1. Perform one of the following steps:
   - In the **Assets** section of the **Start Screen**, click **Playback Visual test '*Project name* - *Test name*'** for the appropriate visual test in the **Recent** list.
   - In the **Asset Browser**, right-click the appropriate visual test and click **Playback**.
   - From the **Visual Navigator**, choose **Actions** > **Playback** to play back the active visual test.

   The **Playback** dialog box opens.
2. *Optional:* Type a description for the test result into the **Result description** field.
3. From the **Result options** list, select how you want to save the results of this test.
   Select one of the following options:

Save results with a new run number	Results from the playback of a visual test or script are saved with a new incremental run number. For example, a new result is saved with a run number of 1. Subsequent results for the same visual test or script are saved with run numbers of 2, 3, 4, and so on. This is the default selection.
Append results to current run number	Results from the current playback are appended to the results of the most recent existing run number. A new run number is not created. This option only appears with an existing result.
Overwrite existing run number with new results	Results from the current playback overwrite the results of the latest existing run number. A new run number is not created. This option only appears with an existing result.
Do not save this result	A result is not created when playing back a visual test or script.

4. *Optional:* Click **Options** to change the result options.
   For additional information, see *Setting Playback Result Options*

5. *Optional:* Check the **Show Local Variables window** check box to display the **Local Variables** window during playback.

   Silk Test Workbench hides the **Local Variables** window when the playback is finished.
6. Click **OK**.
7. If you are testing a web application, the **Select Browser** dialog box opens. Select the browser and click **Run**.

   Note: If multiple applications are configured for the current script, the **Select Browser** dialog box is not displayed.
8. *Optional:* If necessary, you can click **Actions** > **Stop**, click **Stop** on the toolbar, or press both **Shift** keys to stop playback. Click **Actions** > **Pause** or click **Pause** on the toolbar to pause playback at the current step. As the visual test plays back, you can see each action executed against the test application, such as menu, radio button, and check box selections.

   If the visual test contains passed in variables without assigned values, the **Passed in Variables** dialog box opens, where you can assign values to these variables.

   If an error occurs during playback, the **Playback Error** dialog box opens.
9. Once playback completes, use the **Playback Complete** dialog box to determine the action to take.

   Hide the **Playback** dialog box during the playback process by checking the **Do not show this dialog again** check box. This overrides the **Show Playback** dialog option setting in General Options. Use that setting to show the **Playback** dialog box.

Executing a Visual Test Within a Visual Test

Visual tests can call and execute other visual tests and play them back. This allows a modularized approach to application testing and provides greater control over test execution. When executing one visual test within another, it is important to remember to keep test site integrity and ensure any test applications being tested are in the correct initial playback state.

You can use also this procedure to create a driver test.

1. Open the visual test in which you want to insert another test.
2. Select the step after which you want to insert the test.
3. Click **Insert** > **Visual test**. The **Browse for Visual test** dialog box opens.
4. Choose whether you want to display all assets or only assets that you have created.

   Note: Click any column header (**Name**, **Project**, **Description**, or **Modified Date**) to alphabetically sort the list of visual tests by column header type.
5. Select the visual test to insert and then click **OK**.

   Note: Visual tests from the current project and from all referenced projects display in the list.

   Tip: If there are many visual tests in the **Select an asset** list, type the name of the visual test that you want to insert into the **Name Filter** text field and then click on the visual test.

   Silk Test Workbench inserts a step below the selected step. The inserted step calls the selected visual test.

During playback, when the above step executes, the inserted visual test plays back to completion before the next step in the calling visual test executes. Once a visual test has been inserted into another visual test, you can configure the settings used to play it back.

During playback, the tested applications are configured, hooked, and started when required, and remain hooked until the end of the playback session. For example, if the calling visual test has an application configuration for Notepad, and the inserted visual test has an application configuration for Calculator, the following happens:

- Notepad is hooked and started when the execution of the calling visual test starts.
- Calculator is hooked and started when the execution of the inserted visual test starts. Notepad remains hooked.
- Both Notepad and Calculator remain hooked when the execution of the inserted visual test stops.
- Both Notepad and Calculator get unhooked when the playback session ends, which means that the execution of the calling visual test stops.

Visual Test Variable Dialog Box

Use this dialog box to assign values to passed variables that have no value assigned in the visual test being played back. This dialog box only appears during playback, and only if the visual test has parameters or variables defined in its `<<Start>>` step for which there are no values.

Note: Variable names cannot begin with "st_".

The **Visual Test Variable** dialog box contains a list with the following columns:

Visual test Variable Name – Displays the name of the visual test variable in this visual test with no assigned value. This is a variable that is passed in to any other visual test that executes this visual test during its playback. Data in this column is for display only and cannot be changed.

Type – Displays the data type of the visual test variable. Data in this column is for display only and cannot be changed. Variable types include:

Text	The variable's value is a text string. Text type values can contain letters, numbers, spaces, and punctuation. **Note:** The text data type is case sensitive, which means the cases of the individual characters in the text string must match during comparison. If the cases do not match, the comparison fails.
Number (Long)	The variable's value is a number ranging from -2,147,483,648 to 2,147,483,647. Periods (for a fractional/decimal value) cannot be used.
Number (Double)	The variable's value is a double-precision floating-point number, and is stored as a 64-bit number in the range -1.7E308 to +1.7E-307.
Boolean (True/ False)	The variable's value is either `True` or `False`.
Number (Long Long)	A 64-bit integer value that can have a value in the range -9223372036854775808 to 9223372036854775807.
Enumeration	This variable groups together a set of values and orders them sequentially from 1 to *n*. You declare an enumerated type when you want a variable to hold only a limited number of distinct values. Use this variable type with properties or variables that expect an enumeration data type. An enumeration stores the numeric value in a 64-bit integer in the range -9223372036854775808 to 9223372036854775807. However, the engine only supports 32-bit enumerations, while the UI supports 64-bit enumerations (in future releases, the engine will support 64-bit enumerations).

Value – The value for the listed visual test variable. Type a value for the visual test variable. The value must match the type. For example, if the type is Number (Long), the value must be a number between -2,147,483,648 and 2,147,483,647. If the value entered does not match the type, or if no value is entered, a default value is used.

- For Number (Long), Number (Double), Number (Long Long), and Enumeration the default value is 0

- For Text, the default value is a null string
- For boolean, the default value is False

Playing Back Scripts

After you record a script, you can play it back at any time. Code lines in a script are typically executed in sequence from top to bottom. During playback, the script drives the test application according to the actions in the script. You can observe these actions taking place on the screen. The actions are performed at the fastest speed allowed by the application's responses. This allows tests to be performed at speeds much greater than manual execution.

Because scripts are recorded on the object level without reference to an object's position on the screen, playback is also object-oriented. This means that Silk Test Workbench interacts directly with the Windows objects. It does not matter if there have been environmental changes or changes in the test application, scripts still play back properly.

Playing Back a Script

When you playback a script, it performs each of the actions that you have recorded or scripted.

1. Open the script.
2. Choose **Actions** > **Playback** or click **Playback** on the toolbar. The **Playback** dialog box opens.
3. *Optional:* Type a description for the test result into the **Result description** field.
4. From the **Result options** list, select how you want to save the results of this test.

   Select one of the following options:

Save results with a new run number	Results from the playback of a visual test or script are saved with a new incremental run number. For example, a new result is saved with a run number of 1. Subsequent results for the same visual test or script are saved with run numbers of 2, 3, 4, and so on. This is the default selection.
Append results to current run number	Results from the current playback are appended to the results of the most recent existing run number. A new run number is not created. This option only appears with an existing result.
Overwrite existing run number with new results	Results from the current playback overwrite the results of the latest existing run number. A new run number is not created. This option only appears with an existing result.
Do not save this result	A result is not created when playing back a visual test or script.

5. *Optional:* Click **Options** to change the result options.

   For additional information, see *Setting Playback Result Options*
6. *Optional:* Check the **Show Local Variables window** check box to display the **Local Variables** window during playback.

   Silk Test Workbench hides the **Local Variables** window when the playback is finished.
7. Click **OK**.
8. If you are testing a web application, the **Select Browser** dialog box opens. Select the browser and click **Run**.

   Note: If multiple applications are configured for the current script, the **Select Browser** dialog box is not displayed.
9. *Optional:* If necessary, you can choose **Actions** > **Stop**, click **Stop** on the toolbar, or press both **Shift** keys at the same time to stop the script.

Note: If an error occurs during playback, an error message appears.

10. Once playback completes, use the **Playback Complete** dialog box to determine the action to take.

    Hide the **Playback** dialog box during the playback process by checking the **Do not show this dialog again** check box. This overrides the **Show Playback** dialog option setting in General Options. Use that setting to show the **Playback** dialog box.

Showing Automation Actions During Playback

Display automation actions in the **Output** window to review or troubleshoot scripts.

1. Choose **View** > **Output**. The **Output** window opens.
2. Click **Tools** > **Options**. The **Options** dialog box opens.
3. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
4. Click **General**.
5. From the **Show automation actions in the Output window** list box, select **Yes**.

Playback a script to see the related automation actions.

Pausing a Script

You can pause a script at any time during playback by clicking **Pause** on the toolbar.

Running Tests in Parallel

You can use multiple Workbench processes to execute tests in parallel against multiple browsers or mobile devices. For example, you can use this functionality when executing test from a continuous integration server, or from Silk Central.

Silk Testby default supports parallel testing for the following browsers and platforms:

- Google Chrome.
- Mozilla Firefox.
- Web, native, and hybrid apps on the following platforms:
  - Physical Android devices.
  - Android Emulators.
  - Physical iOS devices.

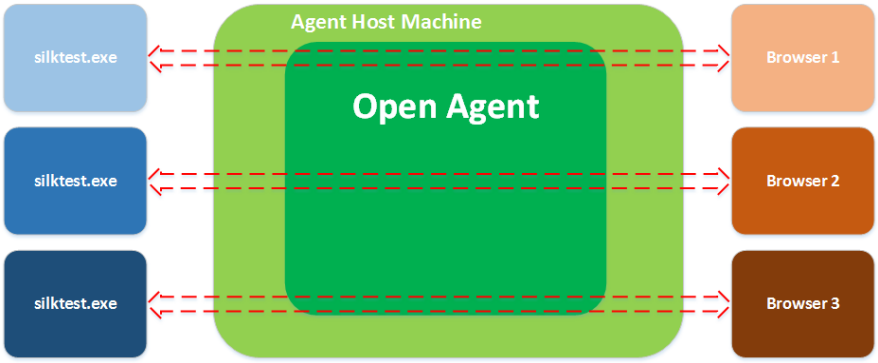
To disable parallel test replay, set the environment variable *SILKTEST_ENABLE_PARALLEL_TESTING* to false.

Note: Enabling parallel testing causes the Open Agent to handle each test-executing process separately. Applications which have been tested in one Silk Test client cannot be tested from another client, while the initial client is running. For example, you cannot test the same application alternating between Silk4J and Silk4NET.

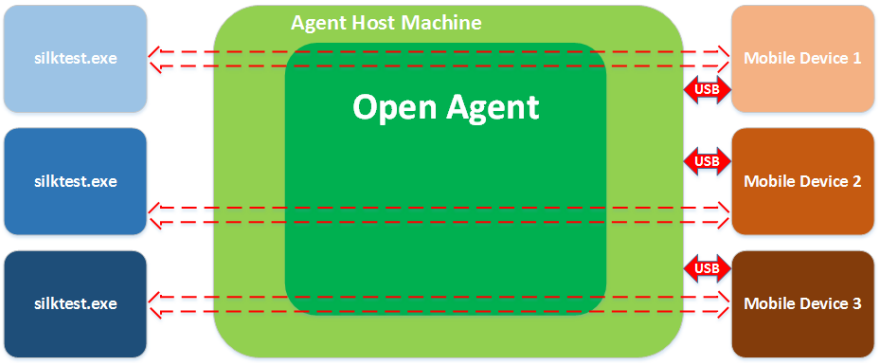
Note: You cannot execute multiple test runs on the same mobile device at the same time. Before running tests in parallel, ensure that enough devices or emulators are available. Any test runs that get no mobile device or emulator assigned will fail.

Each parallel test run starts as a separate silktest.exe, which corresponds to one browser or mobile device. You can specify the browser or mobile device that you want to associate with a specific silktest.exe through the connection string. For additional information, see *Connection String for a Mobile Device* or *Connection String for a Remote Desktop Browser* .

The following image shows testing multiple browsers in parallel:



The following image shows testing multiple devices in parallel:



Multiple processes starting simultaneously might each try to start the Open Agent on the machine on which Silk Test Workbench is running. Running the Open Agent multiple times on the same machine is not possible and will cause Silk Test Workbench to throw an exception. To avoid this, ensure that the Open Agent is running before starting the parallel test runs.

The test results are stored in multiple result files, one for each test run.

Configuring Visual Test Playback

You can configure different aspects of visual test playback behavior such as how the visual test handles verifications, the timing and synchronization of the visual test with the test application, the use of a result and the content that appears in a result, among others.

There are two primary methods for configuring the behavior of visual test playback. The first is to set values for various playback options in the **Options** dialog box. Playback options set in the **Options** dialog box are global options that Silk Test Workbench automatically applies to all subsequent visual test playback. Choose **Tools** > **Options** to set these configuration options.

The second method is to insert a `Set playback setting` step in a visual test and override the global playback option values. Setting playback options in a `Set playback setting` step does not permanently change the default global playback option as it appears in the **Options** dialog box. A `Set playback setting` step only changes the playback option for the visual test in which the step appears.

You can also return the current values of playback options to a variable in a visual test, and use the variable in expressions, test logic, or anywhere a variable can be used.

Setting a Playback Setting Value in a Visual Test

You can set a playback setting in a visual test and override the default value of a global playback option in the **Options** dialog box. Overriding a playback option in a visual test does not permanently change the default value of the playback option as it appears in the **Options** dialog box.

1. Open the visual test in which you want to set the value of the playback setting.
2. Select the step that precedes where you want to insert the playback setting step.
3. Choose **Insert** > **Playback Setting** > **Set Step**. Silk Test Workbench inserts a `Set playback setting` step after the selected step. Properties for the `Set playback setting` step display in the **Properties** pane.
4. Select the `Set playback setting` step to display its properties in the **Properties** pane.
5. In the **Properties** pane, select the desired playback setting and assign its value.

   The following settings are available:

   - All options under **Tools** > **Options** > **Advanced**, with the exception of the options under **Tools** > **Options** > **Advanced** > **ActiveData**.
   - All options under **Tools** > **Options** > **Playback** > **General**.
   - All options under **Tools** > **Options** > **Playback** > **Timing**.
   - All options under **Tools** > **Options** > **Playback** > **xBrowser**.
   - All options under **Tools** > **Options** > **Playback** > **Close**.
   - All options under **Tools** > **Options** > **Playback** > **Playback Status Dialog**.

   Silk Test Workbench updates the label of the Set playback setting step as follows:

   ```
   Set playback setting '[playback setting name]'
   ```
6. Repeat the above steps to create other `Set playback setting` steps as needed. When you playback the visual test, Silk Test Workbench uses the playback setting value specified in the `Set playback setting` step.

Getting a Playback Setting Value in a Visual Test

You can get a playback setting by inserting a step in a visual test that returns the current value of an option and stores the value in a local variable. You can then use the local variable in verification or logic steps or to reset a playback setting value back to its original value.

1. Open the visual test in which you want to get the value of the playback setting.
2. Create a local variable to store the playback setting value.
3. Select the step that precedes where you want to insert the playback setting step.

   Steps that retrieve the value of a setting and set it to a variable must execute sequentially before any steps that use the variable.
4. Choose **Insert** > **Playback Setting** > **Get Step**. Silk Test Workbench inserts a `Get playback setting` step after the selected step. Properties for the `Get playback setting` step display in the **Properties** pane.
5. Select the newly created step to display its properties in the **Properties** pane.
6. Select the appropriate setting and assign its value to a variable.

   The following settings are available:

   - All options under **Tools** > **Options** > **Advanced**, with the exception of the options under **Tools** > **Options** > **Advanced** > **ActiveData**.
   - All options under **Tools** > **Options** > **Playback** > **General**.
   - All options under **Tools** > **Options** > **Playback** > **Timing**.

- All options under **Tools** > **Options** > **Playback** > **xBrowser**.
- All options under **Tools** > **Options** > **Playback** > **Close**.
- All options under **Tools** > **Options** > **Playback** > **Playback Status Dialog**.

Silk Test Workbench updates the label of the Get playback setting step as follows:

```
Get playback setting '[playback setting name]' and set to '[variable name]'
```

7. Repeat the above steps to create other `Get playback setting` steps as needed.

Configuring Playback for Inserted Visual Tests or VB .NET Scripts

Once a VB .NET script or visual test has been inserted into a visual test, you can configure the playback settings applied to the inserted VB .NET script or visual test.

1. Open the visual test that contains the inserted VB .NET script or visual test.
2. Select the step that executes the VB .NET script or visual test.

   For a visual test, the step text looks similar to the following:

   ```
   Playback visual test [asset name]
   ```

   where *[asset name]* is the name of the inserted visual test.
3. Open the **Properties** pane.
4. From the **Playback settings** list, select which settings should be applied during playback to the inserted VB .NET script or visual test.
   - Use system defaults.
   - Inherit settings from parent.
   - Copy settings from parent.

How Does Silk Test Workbench Synchronize Tests?

Many unexpected test failures are related to synchronization issues. Weak synchronization during test replay might generate false negative results, making it difficult to detect actual application problems. Synchronization errors are timing issues that heavily depend on the test environment, making these errors difficult to reproduce and solve. For example, a synchronization error that occurs in a specific test environment might never be reproduced in the development environment. Weak synchronization is one of the most common reasons for an automation project to get unmanageable with a growing set of automated tests.

Silk Test Workbench provides automated *test synchronization* for all supported technologies, enabling you to build robust and manageable test sets. During the replay of a test, Silk Test Workbench ensures that the test always waits until the AUT is ready to perform the next action. For a verification step in a test, Silk Test Workbench ensures that any preceding actions in the test are completed before performing the verification.

To adapt your tests to the specific behavior of your AUT, you can change the values of the following synchronization timeouts:

Synchronization timeout (OPT_SYNC_TIMEOUT)	The maximum time in milliseconds that Silk Test Workbench waits for the AUT to become ready during playback. The default value is 300000 milliseconds.
Object resolve timeout (OBJ_WAIT_RESOLVE_OBJDEF)	The maximum time in milliseconds that the `Find` method searches for an object. The default value is 5000 milliseconds.

Note: To be able to successfully run tests under load or on slow systems, for example a virtual machine accessed through a slow connection, Silk Test Workbench occasionally increases the internal timeout, for example while the AUT is starting and when a dialog or window appears. As soon as the AUT, dialog, or window is completely started, Silk Test Workbench reduces the timeout again to the value you have specified for OPT_WAIT_RESOLVE_OBJDEF.

Object resolve retry interval (OPT_WAIT_RESOLVE_OBJDEF_RETRY)	If Silk Test Workbench cannot immediately find an object, Silk Test Workbench will retry to find the object until the object resolve timeout expires. The object resolve retry interval specifies the time in milliseconds that Silk Test Workbench waits before retrying to find the object. The default value is 1000 milliseconds.
Object enabled timeout (OPT_OBJECT_ENABLED_TIMEOUT)	The maximum time in milliseconds that Silk Test Workbench waits for an object to become enabled during playback. The default value is 1000 milliseconds.

Note: The timeouts do not overlap.

For detailed information about the automated synchronization that Silk Test Workbench provides specifically for Web applications, see *Page Synchronization for xBrowser*. For detailed information about the synchronization that Silk Test Workbench provides specifically for Ajax applications, see *How to Synchronize Test Automation Scripts for Ajax Applications*.

In addition to the automated synchronization, Silk Test Workbench also enables you to manually add wait functions to your VB . NET scripts. Silk Test Workbench provides the following wait functions for manual synchronization:

WaitForObject	Waits for an object that matches the specified locator. Works with an xPath locator or an object map identifier.
WaitForProperty	Waits until the property specified by the `propertyName` parameter gets the value specified by the `expectedValue` parameter or until the timeout is reached.
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached.
WaitForChildDisappearance	Waits until the child object specified by the `locator` parameter does not exist or until the timeout is reached.

If a test randomly fails with an `ObjectNotFoundException`, increase the Object resolve timeout, for example to 30 seconds. For very specific long running operations, for example a click on an object that displays after a long calculation with a progress dialog, manually add the `WaitForObject` method to the test script, to wait until the object is found, or add the `WaitForDisappearance` method to the test script, to wait until the progress dialog is no longer displayed.

Automated synchronization example

Consider the following code sample, where Silk Test Workbench tries to click on a button with the caption **Ok**:

```
'VB .NET code
Dim button = _desktop.PushButton("@caption='Ok'")
button.Click()
```

To replay the actions in this code sample, Silk Test Workbench performs the following synchronization actions:

1. Silk Test Workbench tries to find the button. If the Object resolve timeout runs out, Silk Test Workbench stops the replay and throws an exception.
2. Silk Test Workbench waits until the application under test (AUT) is ready. If the Synchronization timeout runs out before the AUT is ready, Silk Test Workbench stops the replay and throws an exception.
3. Silk Test Workbench waits until the button is enabled. If the Object enabled timeout runs out before the button is enabled, Silk Test Workbench stops the replay and throws an exception.
4. Silk Test Workbench clicks the button.
5. Silk Test Workbench waits until the application under test (AUT) is ready again.

Analyzing Results

A *result* is an asset generated during playback of a visual test or script that provides proof of the testing process and a permanent record of the test application's functional state at any given time. A result contains information about the playback such as the name of the visual test or script, the run number, the date and time each step or line was executed, the pass or fail status of a command or step, and other important types of information. Results provide a valuable way to review playback performance and analyze areas where the test may have failed.

You can view a result in the **Result** window either immediately after playback or by selecting the result from the **Asset Browser**. The **Result** window displays result information using the four panes of the **Visual Navigator**. Additionally, the **Result** window organizes result information using tabs, which provide quick access to a summary report of the playback, any passed or failed verifications, any flagged steps, and the detailed result of every step in a visual test or line of code in a script.

In the **Screen Preview** of a result, you can compare the screens captured during the original recording of a visual test with the screens captured during playback of the visual test. If differences exist, you can update the visual test screen with the playback screen right from the **Screen Preview**. For more information, see *Updating Captured Screens for a Screen Test Step*.

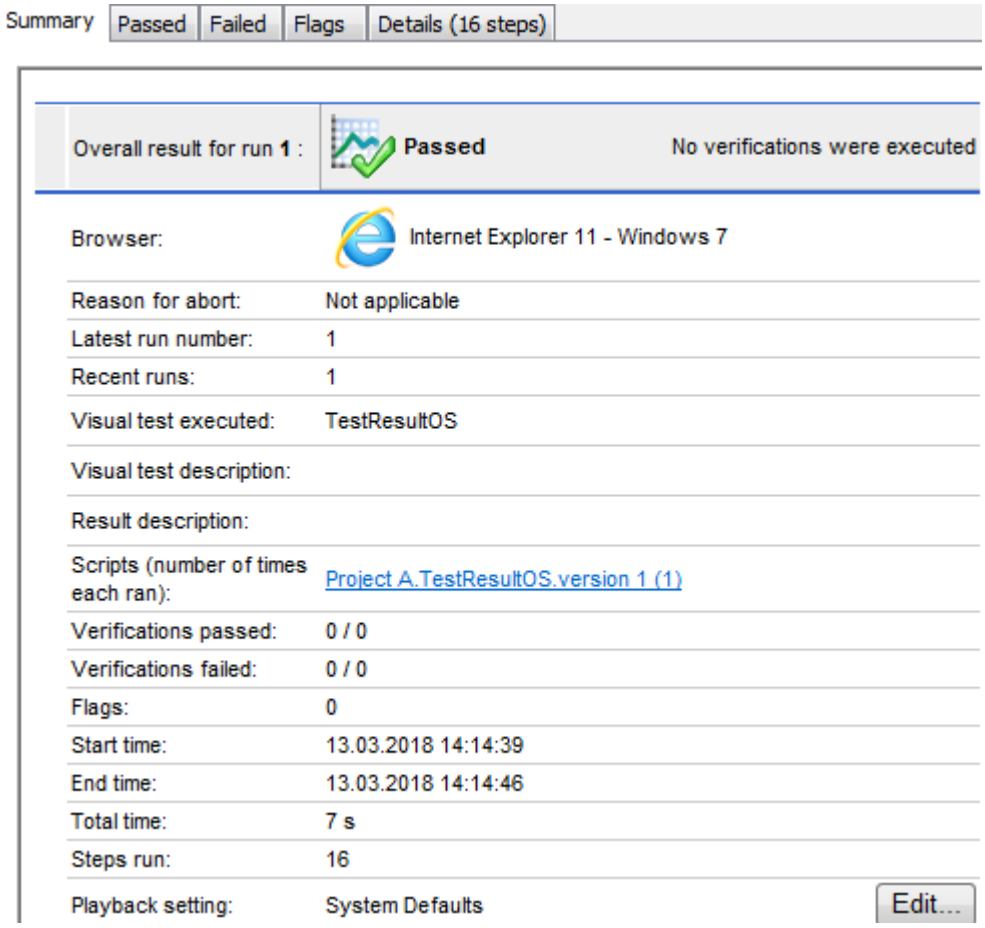
To document your testing procedures or collaborate with other testers, you can print results, save results to a file, or flag items to remind yourself or others about an important issue in a result.

You can also create and apply filters to display only the result information that is of most interest. For example, if you want to view only the failed verifications of the most recent playback, you can create and apply a filter to display only the desired information.

After opening a result, Silk Test Workbench displays the **Result** window from which you can view and analyze detailed information about all aspects of visual test or script playback.

Result Window

After playing back a visual test or script, you can view the results of the playback in the **Result** window.



The **Result** window contains the following features and functionality:

Visual Navigator	Enables you to quickly see all aspects of test playback. For additional information, see *Visual Navigator in the Result Window*.
Result window toolbar	Enables you to easily customize the display and type of content found in a result. For additional information, see *Result Window Toolbar*.
Result window tabs	The tabs organize result content into specific types. For additional information, see *Result Window Tabs*.

Result Window Toolbar

The **Result** window toolbar contains the following buttons and drop-down list for customizing the display and type of content found in a result:

- **Filter Results By Type Selection** – Provides quick access to all pre-defined and user-defined result filters. Select a filter from the list.
- **Manage Filters** – Opens the **Manage Filters** dialog box from which you can create, edit, duplicate, and apply result filters.
- **Criteria** – Displays the **Criteria** dialog box from which you can set a percentage of passed verifications as the criteria to define the success of all future runs. For example, a pass criteria of 90% means that at least 9 out of 10 verifications in a visual test or script must pass for the result of the playback to pass. Setting this option updates the **Result pass criteria (percentage)** option. This percentage is applied to all future results.
- **Show All Runs** – Opens the **Run Detail** dialog box, which lists the results of all test runs of the test, and whether the test run passed or failed. From this dialog box, you can open or delete any result.
- **Refresh** – Refreshes the current result.

- **View** – Sets the type of steps to appear in the **Test Steps** pane. Click the drop-down arrow next to this button and select to view either steps only, screens only, or both. Additionally, you can choose to view the **Step Description** column. The selected view is applied to each tab (**Passed**, **Failed**, **Flags**, and **Details**) in the **Result** window.
- **Basic View** – Displays the standard **Test Steps** pane columns and corresponding properties in the **Properties** pane.
- **Advanced View** – Displays additional columns in the **Test Steps** pane and corresponding properties in the **Properties** pane. To make the additional columns visible in the **Test Steps** pane, you may need to scroll to the right.

Result Window Tabs

The tabs of the **Result** window function as filters that allow you to quickly view specific types of steps of a visual test or code lines of a script.

The **Passed**, **Failed**, **Flags**, and **Details** tabs all organize and display content by using the panes of the **Visual Navigator**. They differ in the type of content found in each tab. The **Details** tab contains every step or code line. The **Passed** tab contains only passed verifications. The **Failed** tab contains only failed verifications. And the **Flags** tab contains only flagged result steps.

The **Result** window contains the following tabs:

Summary tab

When you open a result, this tab is displayed by default. To change the default result view, modify the **Default Result View** option from the General options. The **Summary** tab displays a high-level overview report with the following information:

- **Overall result for run** – Indicates 'Passed' if the visual test or script played back successfully and met the result pass criteria percentage), 'Failed' if it did not meet the result pass criteria percentage, and 'Playback Error' if a step did not perform successfully.
- **Browser** - The following information is displayed in this row if the test was executed against a web application:
  - The browser on which the test was executed.
  - The browser version.
  - The browser size that was used during replay. This is only displayed if you have changed the browser size instead of using the default size.
  - The operating system along with the version.
  - The name of the mobile device, the emulator, or the simulator. This information is only displayed if the test was executed against a mobile web application.
  - The name of the machine to which the mobile device is connected, or on which the emulator or simulator is running, on which the browser test was executed. This information is only displayed if the test was executed against a mobile web application.
- **Device** - The following information is displayed in this row if the test was executed against a native mobile application.
  - The name of the mobile device, the emulator, or the simulator.
  - The operating system of the mobile device, emulator, or simulator, along with the version.
  - The name of the machine to which the mobile device is connected, or on which the emulator or simulator is running.
- **Mobile application** - The full path to the mobile application against which the test was executed. This information is only displayed if the test was executed against a native mobile application.
- **Reason for abort** – Displays the reason playback of a visual test or script was aborted.
- **Latest run number** – Displays the run number of the most current result.

- **Recent runs** – Displays the most recent runs. Click a previous run to view it. To open a previous run not appearing in this field, click **Show All Runs** on the toolbar to open the **Run Detail** dialog box from which you can open or delete any previous run.
- **Visual test/.NET Script executed** – Displays the name of the visual test or script of the result.
- **Visual test/.NET Script description** – Displays the description of the visual test or script of the result.
- **Result description** – Displays the description of the result.
- **Visual tests** or **Scripts** – Lists all the visual tests or scripts that ran successfully as part of the playback, including inserted visual tests or scripts that ran using the `Workbench.RunScript()` method. For example, a driver script could run several scripts in one playback. Additionally displays the version of the visual tests or scripts and the number of times the visual test or script ran.
- **Verifications passed** – The total number of verifications in all visual tests or scripts that executed successfully and passed. Click the number in this field to open **Passed** tab from which you can view all passed verifications.
- **Verifications failed** – The total number of verifications in all visual tests or scripts that executed successfully but failed. Click the number in this field to open **Failed** tab from which you can view all failed verifications.
- **Flags** – For visual tests, the total number of flagged verification steps in the result. Click the number in this field to open the **Flags** tab. Flags are not available for scripts.
- **Start time** – The time the first visual test or script begins playback.
- **End time** – The time the last visual test or script completes playback.
- **Total time** – The total time the visual test or script played back.
- **Steps run** – The total number of steps or code lines run.
- **Playback setting** – Displays the Playback setting which is the group of playback options used to create the result. Click **Edit** to display the Playback options from which you can set the Playback setting.

Passed tab Displays verifications that passed during playback.

Failed tab Displays verifications that failed during playback. Steps that result in a playback error do not appear on this tab.

Flags tab Displays any flagged steps in the result. You can insert a flag in the step of a result by manually inserting it after the result has been created or by using verification logic that inserts a flag during playback of a visual test. The **Flags** tab dynamically displays all flagged steps in the result. For example, after inserting a flag in either of the **Passed**, **Failed**, or **Details** tabs, Silk Test Workbench automatically updates the **Flags** tab to display the flagged step.

Note: Flagged steps in a visual test do not appear in this tab after playback. Only steps flagged in the result and flags inserted by verification logic appear in the **Flags** tab of a result. Flags are not available for scripts.

Details tab Displays information about every step of a visual test or each code line in a script. For example the name of the test step or code line, the pass/fail status, a description, and flag status. For keyword-driven tests, the **Details** tab displays the steps of any keywords that are included in the keyword-driven tests and are implemented as visual tests. The **Details** tab also displays playback details. If an error has occurred during playback, and the visual test or script was called from other visual tests, the playback details include the test steps in the calling visual tests in which the visual test or script was called.

In each of the tabs of the result window, you can select either the **Basic View** or the **Advanced View** of the **Test Steps** pane. The **Advanced View** displays additional columns in the **Test Steps** pane.

Run Status Types

Silk Test Workbench assigns a run status for each test run of a result based on the performance of the test during playback. The run status is displayed in the **Overall result for run** text box on the **Summary** tab of a result.

The run status is dependent on the result pass criteria percentage, which specifies a percentage of passed verifications to define the success of the test run. For example, a pass criteria of 90% means that at least 9 out of 10 verifications in a test must pass for the result of the playback to pass.

Run Status Types

The run status types include:

Type	Description
Passed	A **Passed** run status indicates successful test playback. If a test contains any verifications, a **Passed** status also indicates that the result pass criteria percentage was met.
Failed	A **Failed** run status indicates that a test contains verifications that failed during playback and that the percentage of failed verifications did not meet the result pass criteria percentage.  **Note:** By default, the result pass criteria is set at 100%. To modify this option, click **Criteria** on the **Result** window toolbar.
Playback Error	A **Playback Error** run status indicates that a result contains steps that did not perform successfully. Examples include the following: • Steps that reference a column in an ActiveData file that does not exist • Steps that reference a local variable that does not exist • Steps that run a timer that has not yet been started

For more information about the status of a step in a result, see *Step Status Types*.

Step Status Types

Silk Test Workbench assigns a status for each step in a result based on the performance of the step during playback. The status of a step is displayed in the **Result** column of the **Test Steps** pane and the step text is colored according to its status type.

Step Status Types

Status types for steps in a result include:

Type	Description
Passed	A step is assigned a **Passed** status when a verification step meets the criteria defined by the verification logic of the given step. Steps assigned this status type appear in the **Passed** and **Details** tab of a result. **Passed** appears in the **Result** column and the step text color is green.
Failed	A step is assigned a **Failed** status when a verification step fails to meet the criteria defined by the verification logic of the given step. Steps assigned this status type appear in the **Failed** and **Details** tab of a result. **Failed** appears in the **Result** column and the step text color is red.
Playback Error	A step with a **Playback Error** status is a step that does not perform successfully. Examples include the following: • Steps that reference a column in an ActiveData file that does not exist

Type	Description
Error Screen	• Steps that reference a local variable that does not exist • Steps that run a timer that has not yet been started **Playback Error** appears in the **Result** column and the step text color is blue. A step that appears after a step with **Playback Error** status. A step with **Error Screen** status displays the screen of the test application as it appeared when the test incurred a playback error. **Error Screen** appears in the **Result** column and the step text color is blue. **Note:** This step type is not applicable to test scripts.
Step Performed Successfully	Any step (other than a verification step) that performs successfully. The **Result** column is empty and the step text color is black.

For more information about the run status of a result, see *Run Status Types*.

Visual Navigator in the Result Window

The **Result** window contains the Visual Navigator, which allows you to quickly see all aspects of test playback.

For visual tests, the **Passed** tab, **Failed** tab, **Flags** tab, and **Details** tab all contain the four panes of the Visual Navigator:

- Test Steps
- Screen Preview
- Properties
- Storyboard

Each of the **Passed**, **Failed**, **Flags**, and **Details** tabs of the **Result** window organize and display content using the four panes of the Visual Navigator. The **Test Steps**, **Screen Preview**, **Storyboard**, and **Properties** panes are synchronized with each other and display information specific to a selected step in the **Test Steps** pane. In the **Result** window, the Visual Navigator panes contain additional functionality which is described in the following sections.

Test Steps

When viewing a result, the **Test Steps** pane displays the same information as it does for a visual test with additional data for the result of each test step. You can select a step and update the other panes with information specific to the selected step. Additionally, you can insert a flag, or add a step description to a selected step. You also have the option of displaying results in the Basic view or Advanced view.

Screen Preview

In the screen preview of a result, you can compare the screen captured during recording of a visual test against the screen captured during playback. Select the step in the **Test Steps** pane that contains the screen to compare, and then from the **Screen Preview**, click **Actions** > **Show Differences** > **Side by Side**. If you want to update the visual test to contain the screen captured during playback, click **Actions** > **Update Screen**.

Storyboard

The **Storyboard** pane of a result uses icons to indicate the execution status of a verification logic step.

Properties

The **Properties** pane for a result displays the properties of the result and the values captured during playback.

The **Properties** pane contains a **Show/Hide Step Properties of Visual Test Before Playback** toolbar button, which allows you to show or hide the visual test properties of a selected step.

For scripts, the **Result** window displays only the **Test Step** and **Properties** panes. The **Test Steps** pane contains additional columns which provide more information about the playback status and the result of each test step.

To customize the display of result data, you can modify the **Default result view** from the **General** options. You can also modify playback results options and create filters to view desired data.

You can display specific properties in the **Properties** pane by using the Basic view or Advanced view options from the **Result** window toolbar.

Managing Results

By default, Silk Test Workbench creates a result after playback of a visual test or script and saves the result as a unique asset. To view a result, you can use any of the following methods:

- Manually open a result after playback from the **Playback Complete** dialog box
- Automatically open a result immediately after playback
- Open a result from the asset browser at a later time

After opening a result, the **Result** window appears and displays the **Summary** tab of the result by default. To change the default result view, modify the **Default Result View** option from the **General** options.

The default name of a result is the name of the visual test or test script being played back. You can modify the **Default Result Name** option in the **Playback Results** options.

Creating a Result

1. Open a visual test or script.
2. Click **Playback**. The **Playback** dialog box opens.
3. In the **Result description** text box, enter a description of the result.
4. Select how you want to save the result by selecting an option from the **Result option** list.

   Choose one of the following options:

   - **Save results with a new run number** – Results for the playback are saved with an incremental run number. This is the default selection.
   - **Append results to current run number** – Results from the current playback are appended to the results of the latest existing run number. A new run number is not created. This setting only appears with an existing result.
   - **Overwrite existing run number with new results** – Results from the previous playback are overwritten with the results of this playback, using the run number from the previous playback. This setting only appears after the first time the visual test or script is played back.
   - **Do not save this result** – Play back without creating a results file. If **Append results to current run number** or **Overwrite existing run number with new results** is selected for a visual test or script that has no existing result, then **Save results with a new run number** is used.
5. Click **OK**.

Customizing the Behavior and Appearance of a Result

Silk Test Workbench provides several different methods for configuring the behavior and appearance of a result.

You can set values for various playback options, including result options, in the **Options** dialog box. Playback options set in the **Options** dialog box are global options that Silk Test Workbench automatically applies to all subsequent visual test playback. For more information, see *Modifying Playback Options*.

For visual tests, you can override the default result options by inserting a `Set playback setting` step and selecting the desired playback setting. For more information, see *Setting a Playback Setting Value in a Visual Test*.

For both scripts and visual tests, you can also control some aspects of result recording directly from the **Playback** dialog box. For more information, see *Turning Result Recording On or Off*.

Turning Result Recording On or Off

By default, Silk Test Workbench creates a result for each playback of a visual test or script. You can prevent Silk Test Workbench from creating a result after playback by setting the **Result option** in the **Playback Results** options or in the **Playback** dialog box, which appears by default when you playback a visual test or script.

1. Choose **Tools** > **Options**. The **Options** dialog box opens with the **General** options listed by default.
2. Double-click the **Playback** option from the tree view and click **Results**. The right pane of the dialog box displays the available result options.
3. To turn off result recording, select **Do not save this result** from the **Result option** list.
4. To turn on result recording, select one of the following options from the **Result option** list.
   - **Save results with a new run number** – Results for the playback are saved with an incremental run number. This is the default selection.
   - **Append results to current run number** – Results from the current playback are appended to the results of the latest existing run number. A new run number is not created. This setting only appears with an existing result.
   - **Overwrite existing run number with new results** – Results from the previous playback are overwritten with the results of this playback, using the run number from the previous playback. This setting only appears after the first time the visual test or script is played back.
5. Click **OK**.

Opening a Result Manually After Playback

To view a result, you can open it manually after playback.

1. Choose **Tools** > **Options**. The **Options** dialog box opens with the **General** options listed by default.
2. Verify or set the **Show Playback Completed dialog** option to **Yes**.
3. Run your visual test or script. The **Playback Complete** dialog box opens.
4. Click **Go to Result**. The **Result** window opens and displays the result.

Opening a Result Automatically After Playback

To view a result, you can set Silk Test Workbench to automatically open results.

1. Choose **Tools** > **Options**. The **Options** dialog box opens with the **General** options listed by default.
2. From the **Open result automatically** list box, select **Yes**.
3. From the **Show Playback Competed dialog** list box, select **No**.

   This prevents the **Playback Complete** dialog box from appearing after playback.
4. Run your visual test or script. After playback, Silk Test Workbench automatically opens the **Result** window and displays the result.

Viewing a Result From the Asset Browser

To view a result, you can open a result from the asset browser at a later time.

1. Choose **View** > **Asset Browser**. The **Asset Browser** window opens.
2. From the **Asset Types** list, select **Result**. The items pane displays the list of existing results.
3. Double-click the result you want to view. The **Result** window opens and displays the result.

Tracing Errors in Multiple Visual Tests

When a visual test or a VB .NET script is called from another visual test, and an error occurred during execution, the **Details** tab in the result of the calling visual test displays the test step in which the error occurred, as well as the test step or line of code in the respective called visual test or VB .NET script, in which the error occurred. If the calling visual test was also called by a step in another visual test, the result of the calling visual test additionally displays the step in the calling visual test and so on. To see these test steps, perform the following actions:

1. Open the result of the visual test that was executed.
2. Select the **Details** tab.
3. Select the step in which the error has occurred.
4. Open the **Properties** pane The **Called From** list in the **Playback details** section lists the step in which the error has occurred in the calling visual test and in each called visual test or test script, as well as the name of the corresponding visual test or test script.
5. Select an asset from the list and click **Open Asset** to directly access the step or line in which the error occurred.

Example

If the second step of the visual test *Test1* calls the visual test *Test2*, and the fifth step of **Test2** calls the visual test *Test3*, and an error occurs in the fourth step of *Test3* during the execution of *Test1*, the result file for *Test1* includes the following entries:

Called From	
Test1	2
Test2	5
Test3	4

Switching Between the Result Window and Visual Test Window

View a visual test that is linked to a specific result.

1. Open the result of a visual test.
2. Click the **Details** tab, and then click **Actions** from the **Test Steps** pane title bar.
3. Choose **Visual test Window** .
4. To return to the **Result** window, click **Actions** from the **Test Steps** pane title bar of the visual test, and then choose **Go To Result Window**.

Inserting a Result Comment in a Visual Test

Result comments allow you to provide supplemental information about a visual test in the result. After playback of a visual test, the comment appears as a step in the **Details** tab of the **Result** window.

Tip: In addition to inserting a result comment, you can also use a flag to insert a comment on a selected step.

1. For a visual test, in the **Test Steps** pane of a visual test, select the step that you want the result comment to appear after.

2. Choose **Insert** > **Result Comment**. A `Result Comment` step appears in the **Test Step** pane and properties for this step appear in the **Properties** pane.
3. In the **Properties** pane, perform one of the following steps:
   - Type a comment in the **Expression** property value text box.
   - Click the text box to enable the **Select** button from which you can select a variable or create an expression to use as the comment value.
4. Play back the visual test. From the **Details** tab of the result, you can view the comment in the **Result Detail** column of the `Result Comment` step.

Printing a Visual Test Result

1. Open the result that you want to print.
2. Choose **File** > **Print**. The **Print** dialog box opens.
3. Select the printer from the list.
4. *Optional:* Click **Properties** to change the page setup settings, for example margins and orientation.
5. *Optional:* In the **Page Range** section, specify whether to print the entire document or only selected pages.
6. *Optional:* In the **Number of copies** filed, specify the number of copies to print.
7. *Optional:* Check the **Collate** check box to print the corresponding page of every copy together.

   This check box is only enabled when you have selected to print more than one copy.
8. To customize the print output, click **Options** and select the type of result information that you want to include in the printed output.

   Options include:

Include — Specifies the type of summary to include.

- **Visual test summary** – The summary contains the name, project, creator, creation date, version, and description of the visual test. This summary appears on the last page of the printed output.
- **Result summary** – Includes the summary page of the latest result. This summary appears on the first page of the printed output.
- **Group screens** – Includes the group screen snapshots, which contain the main context of the current dialog box or Web page, in the printed output. The test steps that follow the group screen correspond with a single action that occurs on the context of the current dialog box or Web page. The group screen corresponds with the `Using` step in the **Test Steps** pane. The group screen snapshots appear in the printed output as they do in the **Screen Preview** pane.

Tabs — Specifies the tabs from which to print content. In the printed output, content from the **Details** tab appears first, followed by the **Passed** tab, **Failed** tab, and **Flags** tab content.

- **Passed** – Includes the contents from the **Passed** tab of a result.
- **Failed** – Includes the contents from the **Failed** tab of a result.
- **Flags** – Includes the contents from the **Flags** tab of a result.
- **Details** – Includes the contents from the **Details** tab of a result.

Include for each step — Specifies the content to include for each step.

- **Screen** – Includes any associated screen snapshots for each individual step. Individual screens correspond with a `(step screen)`, which is associated with a specific action on the current dialog box or Web page. Step screen snapshots appear in the printed output as they do in the **Screen Preview** pane.

- **Description** – Includes the step description for each step. The step description in the printed output appears as it does in the **Step description** property of the **Properties** pane.
- **Flag details** – Includes flag details such as the flag description, creator, modifier, modification date, and assigned user.

Include the following properties

Specifies the properties to include for each step.

- **Extended properties** – Includes the extended properties, which provide supplemental information about a step.
- **All other properties** – Includes all properties except for extended properties of each step.

Default

Click this button to select the default print options.

9. *Optional*: Click **Preview** to display how the result information will look when printed. The **Preview** window opens.
10. Click **OK**.

Printing a Script Result

1. Open the result that you want to print.
2. Choose **File** > **Print**. The **Print** dialog box opens.
3. Set the print options.
4. To customize the print output, click **Options** and select the type of result information that you want to include in the printed output.

   Options include:

Include

Specifies the type of content to include.

- **.NET Script summary** – Includes a summary of the script that contains the name, project, creator, creation date, version, and description of the script. This summary appears on the last page of the printed output.
- **Result summary** – Includes the summary page of the latest result of the script. This summary appears on the first page of the printed output.
- **Group screens** – Not available for scripts.

Tabs

Specifies the tabs from which to print content. In the printed output, content from the **Details** tab appears first, followed by the **Passed** tab, **Failed** tab, and **Flags** tab content.

- **Passed** – Includes the contents from the **Passed** tab of a result.
- **Failed** – Includes the contents from the **Failed** tab of a result.
- **Flags** – Includes the contents from the **Flags** tab of a result.
- **Details** – Includes the contents from the **Details** tab of a result.

Include for each step

Specifies the content to include for each step.

- **Screen** – Includes any associated screenshots for each individual step. Individual screens are associated with a specific action against a control in the application under test.
- **Description** – Not available for scripts.
- **Flag details** – Includes flag details such as the flag description, creator, modifier, modification date, and assigned user.

Include the following properties	Specifies the properties to include for each step. • **Extended properties** – Not available for scripts. • **All other properties** – Includes all properties for the script.
Default	Click this button to select the default print options.

5. *Optional*: Click **Preview** to display how the result information will look when printed. The **Preview** window opens.
6. To change the margins, orientation, and other page setup information, click **Properties**.
7. Click **OK**.

Saving Results to a File

To document your testing procedures or collaborate with other testers, you can save the result of a visual test or test script to a file in either XML or tab-delimited, plain text format. After saving the file, you can then import the file into a reporting tool to view, manage, format, or print.

Note: To quickly view the result of a visual test or test script saved as a text file, you can import the file into Microsoft Excel as a tab-delimited file. The results appear organized by column header.

1. Open the result of a visual test or script.
2. Click one of the following results tabs:
   - **Summary**
   - **Passed**
   - **Failed**
   - **Flags**
   - **Details**

   Only data that appears for a selected tab is saved to the specified file. Filtered data that does not appear and data existing in other result tabs is not saved.
3. Choose **File** > **Save As**. The **Save As** dialog box opens with the default result name listed in the **File name** text box.

   Note: To change the default result name, modify the **Default Result Name** option in the **Playback Results** options.
4. Select a location in which to save the file.
5. Enter a file name or use the default name.
6. Select a type of file format.

   You can save the results of any tab in either an XML or tab-delimited, plain text format.
7. Click **Save**.

Capturing the Contents of a Web Page

To capture a screenshot of the the part of the web page that is currently visible in the browser window, you can use the `CaptureBitmap` method. You have to specify the absolute or relative file path to the location and the name for the image file as a parameter. For example:

```
'VB code
browserWindow.CaptureBitmap("C:\Temp\MyPage.png")
```

To capture a screenshot of the entire contents of a web page as a single image, you can use the `CaptureFullPageBitmap` method. You have to specify the absolute or relative file path to the location and the name for the image file as a parameter. For example:

```
'VB code
browserWindow.CaptureFullPageBitmap("C:\Temp\MyPage.png")
```

Result Filters

Results often contain a great deal of data. To help you focus on the most important data, you can apply pre-defined filters or define your own filters based on criteria such as pass/fail status, date/time, and flagged steps. Silk Test Workbench applies filters to the **Passed**, **Failed**, **Flags**, and **Details** tabs in the **Result** window regardless of what view is currently visible.

Creating a Result Filter

Result filters reduce the amount of data that displays in the **Result** window and help you quickly find important results. Once you have set up a filter, apply it by selecting it from the **Filters** list.

1. Open a result.
2. Click **Manage Filter** on the toolbar. The **Manage Filters** dialog box opens.
3. Click **New**. The **New Filter** dialog box opens with the **General** tab selected by default.
4. In the **Name** text box, type the name for the filter.

   This is the name that appears in the **Filter** list on the toolbar.
5. *Optional*: In the **Description** text box, enter a description about the use of this filter.
6. Select a filter type.

   - Select **Personal** to make this filter available to the current user only.
   - Select **Public** to make this filter available to all users.
7. In the **Filter by** section of the dialog box, select one of the following options from the **Results** list:

All	Displays all the steps in a visual test or code lines in a script.
Failed	Displays only those steps or code lines that have not executed successfully.
Passed	Displays only those steps or code lines that have executed successfully.
8. Select a **Date/Time** option.

   - Select **Any** to view result data that occurred at any time of any day.
   - To view result data that occurred during a specific time or day, select **From** and specify the desired date and time range.
9. Select a **Users** option.

   - Select **All** to view result data created by all users.
   - Select **Selected**, and then select the desired users to view result data created only by the selected users.
10. *Optional*: Check the **Only show steps with flags assigned** check box to filter out all result data that does not have an assigned flag.
11. To create a more precise filter, click the **Advanced** tab, and then select only the desired verifications to include in the filter.

    By default, Silk Test Workbench includes all verifications in the filter.

    In the left pane, numbers that display in parentheses after a command type refer to the number of selected commands out of the total number of commands available. For example, **Verifications (3/7)** means that you selected three verification types out of seven verification types available. Additionally, a bold check indicates that all commands are selected for a given command type. Depending on your operating system settings, a square inside the check box or disabled check indicates a partial selection of commands for a given command type.
12. Click **OK**. The **Manage Filters** dialog box opens.
13. To apply the filter to the active **Result** window, select the filter, click **Apply** and **OK**.

14. Click **Close**.

Editing a Result Filter

Edit a filter to modify the data that is displayed in the **Result** window.

1. Open a result.
2. Click **Manage Filter** on the toolbar. The **Manage Filters** dialog box opens.
3. Select a user-defined filter.

   You cannot edit the default filters (**Verifications** and **Failed Verifications**).
4. Click **Edit**. The **Edit Filter** dialog box opens with the **General** tab selected by default.
5. In the **Name** text box, type the name for the filter.

   This is the name that appears in the **Filter** list on the toolbar.
6. *Optional*: In the **Description** text box, enter a description about the use of this filter.
7. Select a filter type.
   - Select **Personal** to make this filter available to the current user only.
   - Select **Public** to make this filter available to all users.
8. In the **Filter by** section of the dialog box, select one of the following options from the **Results** list:

All	Displays all the steps in a visual test or code lines in a script.
Failed	Displays only those steps or code lines that have not executed successfully.
Passed	Displays only those steps or code lines that have executed successfully.
9. Select a **Date/Time** option.
   - Select **Any** to view result data that occurred at any time of any day.
   - To view result data that occurred during a specific time or day, select **From** and specify the desired date and time range.
10. Select a **Users** option.
    - Select **All** to view result data created by all users.
    - Select **Selected**, and then select the desired users to view result data created only by the selected users.
11. *Optional*: Check the **Only show steps with flags assigned** check box to filter out all result data that does not have an assigned flag.
12. To create a more precise filter, click the **Advanced** tab, and then select only the desired verifications to include in the filter.

    By default, Silk Test Workbench includes all verifications in the filter.

    In the left pane, numbers that display in parentheses after a command type refer to the number of selected commands out of the total number of commands available. For example, **Verifications (3/7)** means that you selected three verification types out of seven verification types available. Additionally, a bold check indicates that all commands are selected for a given command type. Depending on your operating system settings, a square inside the check box or disabled check indicates a partial selection of commands for a given command type.
13. Click **OK**. The **Manage Filters** dialog box opens.
14. To apply the filter to the active **Result** window, select the filter, click **Apply** and **OK**.
15. Click **Close**.

Applying a Result Filter

There are two methods for applying a result filter. You can apply a filter to only the active **Result** window or you can apply a result filter to all opened **Result** windows.

1. Open a result.
2. To apply the result filter to only the active **Result** window, select a filter from the **Filter Results By Type Selection** list from the **Result** window toolbar. Silk Test Workbench applies the filter to the active **Result** window and updates the title bar with the name of the filter. For example: Silk Test Workbench - [Common - VisualTest1 (Result: Filter = Only Failed Verifications)]
3. To apply a result filter to all open **Result** windows, perform the following steps.
   a) Open a result.
   b) Select the filter that you want to apply.
   c) Click **Apply**, and then click **OK**. Silk Test Workbench applies the filter to the active **Result** window and updates the title bar with the name of the filter. For example: Silk Test Workbench - [Common - VisualTest1 (Result: Filter = Only Failed Verifications)]

   After creating a result filter, it appears in the drop-down list immediately to the left of the **Manage Filter** toolbar button on the **Result** window toolbar. When you select a filter from this list, Silk Test Workbench applies it to all opened **Result** windows and updates the title bar with the name of the filter.

Duplicating a Result Filter

Use the **Manage Filters** dialog box to create a copy of a result filter.

1. Open a result.
2. Click **Manage Filter** on the toolbar. The **Manage Filters** dialog box opens.
3. From the **Filters** list, select the filter that you want to duplicate.
4. Click **Duplicate**. The **Duplicate Filter** dialog box opens.

Removing an Applied Result Filter

Remove result filters to display all the result data in the **Result** window.

1. Open a result.
2. Click **Manage Filter** on the toolbar. The **Manage Filters** dialog box opens.
3. Select **(None)**.
4. Click **Apply**.
5. Click **Close**. The filter is removed from the active **Result** window.

Deleting a Result Filter

Delete a user-defined filter to display results with no filters applied.

1. Open a result.
2. Click **Manage Filter** on the toolbar. The **Manage Filters** dialog box opens.
3. Select the user-defined filter to delete.

   You can only delete user-defined filters. You cannot delete the pre-defined filters.
4. Click **Delete** and then click **Yes** to confirm that you want to delete the filter.
5. Click **Close**. The filter is deleted from the current user's Silk Test Workbench database and the default filter **None** is applied to all open results.

Criteria Dialog Box

The **Criteria** dialog box allows you to set a user-defined percentage of passed verifications as the criteria to define the success of all future test runs.

This dialog box contains the following options:

Pass criteria – Set a percentage of passed verifications as the criteria to define the success of all future runs. For example, a pass criteria of 90% means that at least 9 out of 10 verifications in a visual test must pass for the result of the playback to pass.

Setting this option also sets the **Result pass criteria** option in playback results.

Default – Click this button to reset the pass criteria to the default value (100%).

Enabling the Playback Status Dialog Box

Enable the **Playback Status** dialog box to view the actions that are performed during the replay of a test. This dialog box is very useful when replaying tests on another machine, for example on a remote Mac or on a mobile device.

To enable the **Playback Status** dialog box:

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. .
3. Click **Playback Status Dialog**.
4. To enable the **Playback Status** dialog box, set the **Show Playback Status Dialog** option to **Yes**.
5. To display a video or screenshots of the application under test in the **Playback Status** dialog box, set the **Display application under test** option to **Yes**.
6. Click **OK**.

Note: If you are testing on a remote Mac or on a mobile device, ensure that the Mac or device does not switch off the screen during testing, otherwise the **Playback Status** dialog box will not display anything.

Playback Complete Dialog Box

Use this dialog box to determine the action to take after playback completes successfully.

This dialog box does not appear after successful playback if the **Do not show this message again** check box has been previously checked, or the **Show Playback Complete** dialog option in **General** options has been set to **No**. In this case, a visual test displays in the **Visual Navigator**, and a script displays in the script window upon successful playback completion.

The **Playback Complete** dialog box contains the following:

Playback Again – Click this button to play back the visual test or script. When using this selection, the **Playback Completed** dialog box displays again after successful playback.

Go to Result – Click this button to open the **Result** window. For more information on viewing and working with results, see *Results Overview*.

Go to Visual test – Displays only after playing back a visual test. Click this button to open the current visual test in the **Visual Navigator** without saving any changes.

Go to .NET Script – Displays only after playing back a script. Click this button to open the current test script in the script window without saving any changes.

Save – Click this button to view the current visual test or test script after saving any changes.

Do not show this message again – Check this check box to prevent the **Playback** dialog box from appearing for future playbacks. This option can be selected again by setting the **Show playback completed dialog** in the **General Options** to **Yes**.

Tip: If the **Do not show this message again** check box has been checked in the **Recording Completed** dialog box, this dialog box does not display. Instead the **Visual Navigator** opens when a visual test is recorded and the script window displays for a script.

Playback Error Dialog Box

Use this dialog box to determine the action to take when a playback error occurs.

This dialog box appears when Silk Test Workbench encounters an error during visual test or script playback.

The **Playback Error** dialog box contains the following buttons:

Continue – Resumes playback by retrying the step that caused the error. The error appears in the **Details** tab of the playback result for the visual test. Use this option to resume playback after resolving the error. This button is enabled for visual tests only.

End – Stops playback at the step causing the error. The **Playback Complete** dialog box opens, allowing you to replay the visual test or script, display the results for the playback, or return to the visual test or script.

Debug – Enters debug mode, which allows stepping through playback to help diagnose and fix errors in the visual test. This button is enabled for visual tests only.

Help – Displays this help topic in the online help.

Managing Assets

Assets are the basic building blocks of a test project. An asset is any testing element defined and stored in the database. You can view and manage assets using the **Asset Browser**.

Use toolbar buttons and the **Asset Browser** to manage assets. Navigate to the **Get Started** pane in the **Start Screen** and click **Asset Browser** in the **Tasks** area of the pane. The **Asset Browser** provides a single point for creating, managing, and viewing assets in test projects.

You can re-use or redefine most assets without changing other assets that refer to them.

Asset Types

Silk Test Workbench provides support for the following asset types:

Visual Test Visual tests are automated tests that use screenshots to mimic the actions of a user while testing an application. Visual tests comprise the basic building blocks of an automated testing solution.

For information on how to create visual tests, see *Creating Visual Tests*.

.NET Script VB .NET scripts are also automated tests that mimic the actions of a user while testing an application. These scripts are written in Microsoft Visual Basic .NET.

For information on how to create VB .NET scripts, see *Creating Scripts*.

Keyword Test A *keyword-driven test* is an executable collection of keywords. A keyword-driven test can be played back just like any other test.

For additional information about keyword-driven tests, see *Keyword-Driven Tests*.

Keyword A *keyword* is a defined combination of one or more actions on a test object. The implementation of a keyword can be done with various tools and programming languages, for example Java or .NET. In Silk Test Workbench, a keyword is a visual test or a method with the attribute Keyword before the method name. Keywords are saved as keyword assets.

A *keyword sequence* is a keyword that is a combination of other keywords. Keyword sequences bundle often encountered combinations of keywords into a single keyword, enabling you to reduce maintenance effort and to keep your tests well-arranged.

For additional information about keywords, see *Keywords*.

Result A result is an asset generated during playback of a visual test or a .NET script. A result provides proof of the testing process and a permanent record of the functional state of the test application at any given time.

For additional information about results, see *Analyzing Results*.

Object Map Object maps store items that associate logical names with controls and windows. You can then use these logical names, instead of using locators, to identify the controls and windows in your application.

Note: When you rename an object map that includes object map items, the root item name is changed. Any existing visual tests or scripts that use this object map must be manually changed to use the new object map root item name.

For additional information about object maps, see *Object Maps*.

Image Verification	An image verification is a check of whether an image exists or not in the UI of the application under test. For additional information about image verifications, see *Image Verifications*.
ActiveData	ActiveData is used by visual tests to create data-driven tests. ActiveData enables you to leverage existing data in external files as input for powerful, comprehensive application-testing solutions. You can perform multiple transactions against test applications using a different set of data for each transaction without writing complicated code or compromising existing data. For additional information on ActiveData, see *ActiveData*.
Image Asset	An image asset is an image that is stored as an asset in the database. You can use an image asset to click on a control that is otherwise not recognized by Silk Test Workbench. For additional information on image assets, see *Image Assets*.

Asset Version Purge

Silk Test Workbench creates a new version of an asset each time it is saved. Therefore, the number of asset versions may become large and increase the size of the database.

To delete asset versions from the database, use the **Purge Asset Versions** dialog box to specify criteria such as the type of asset, the project in which an asset belongs, the number of versions to keep, and the modified date of an asset.

Note: The current version of an asset cannot be deleted. This feature is not designed to completely delete assets, only versions of assets. You must have Script Writer permissions or Full Access permissions to purge asset versions.

Purging Asset Versions

Before trying to purge asset versions, ensure that your access level corresponds to the minimum access level that is required. To change the minimum access level in the global options, you require administrator rights. For additional information on changing the minimum access level, see *Global Options*.

1. Click **File** > **Purge Asset Versions**. The **Purge Asset Versions** wizard appears.
2. Click **Next**.
3. Select how you would like to purge assets:
   - To purge multiple assets of a specific type, click **Purge by Asset Type**.
   - To purge individual assets, click **Purge Individual Assets**.
4. If you have selected to purge multiple assets of a specific type, perform the following actions:
   a) Check the asset types from which you want to purge asset versions.
   b) Check the projects from which you want to purge versions of the selected asset types.
5. If you have selected to purge individual assets, perform the following actions:
   a) In the projects tree, expand the project node which contains the assets from which you want to purge versions.
   b) Click the asset type.
   c) Check the individual assets from the list.
6. *Optional:* To persist the projects that are selected in the **Purge Asset Versions** wizard, set the **Restore Selected Projects in Purge Asset Versions** option.
   a) Click **Tools** > **Options** in the menu.

b) In the **Options** menu tree, select **Global** > **Asset Management**.
c) Select **Yes** from the **Restore Selected Projects in Purge Asset Versions** list.

When the option is set to **No**, no projects are selected when the **Purge Asset Versions** wizard opens.

7. Click **Next**.
8. Select whether you want to keep a specific amount of asset versions, or whether you want to delete all asset versions modified prior to a specific date.
9. If you want to keep a specific amount of asset versions, specify the number of asset versions to keep.

   Asset version that are older than the selected number are deleted from the database. For example, if you select 2, the latest two versions of the asset are kept and the rest are deleted. The minimum number of versions you can keep is 1 and the maximum number of versions is 32767.

   Note: All actual checks are kept during purge and are deleted when results are deleted. Expected checks that are in use by results are also not deleted. After a purge, more expected checks that specified might be retained. For example, if 5 versions of an expected check are referenced by test results and you want to leave only 3 versions for each asset after the purge, all 5 versions of the check that are referenced by results are retained.
10. If you want to delete all asset versions modified prior to a specific date, select a predefined date from the list or choose a custom date between `01-01-1970` and `12-31-3000`.

    Note: All actual checks are kept during purge and are deleted when results are deleted. Expected checks that are in use by results are also not deleted. This might lead to situations where expected checks are retained which are older than the date you have specified.
11. Click **Next**. The **Purge Asset Versions** wizard shows a summary of the asset versions that are going to be deleted.
12. Click **Next**.

    Note: If an asset or multiple assets could not be purged, a message box will indicate the number of assets that could not be purged. This is usually due to another user having the current version of an asset opened or the asset has been deleted while the purge is occurring.
13. Click **Finish**.

Purge Command Line Parameters

As an alternative to using the GUI, you can use the command line to purge asset versions.

The command line program is named `STWPurge.exe` and is located in: `\Silk\SilkTest\ng\gui\`.

Parameter	Syntax	Example	Description
Username	`u` `username`	`-u admin` `/username JohnSmith`	Specifies the name of the user authorized to connect to the database. This parameter is required.
Password	`p` `password`	`-p admin123` `/password admin123`	Specifies the case sensitive password of the user authorized to connect to the database. This parameter is required unless the password is `Nothing`.
DSN	`d` `dsn`	`-d mydsn` `/dsn mydsn`	Specifies the name of the Data Source Name (DSN) used to connect to the database. This parameter is optional. If omitted, the default DSN is used.
Asset Types	`s` `assettypes`	`-s visualtest script`	Specifies the asset type(s) from which to delete asset versions. This parameter is optional. If omitted, all asset types are specified. This parameter accepts the

Parameter	Syntax	Example	Description
		`/assettypes visualtest`	following values (syntax is indicated in brackets; multiple values are separated by a space) : • Visual test [visualtest] • Object Map [objectmap] • Result [result] • Active Data [activedata] • .NET scripts [script]

Projects	`r`
	`prj`
	`projects`
		`-r commonprojectA`
		`/projects common`
			Specifies the project, the asset type, or the individual asset from which to delete asset versions. This parameter is optional. If omitted, all projects are specified. Multiple entries must be separated by a space. Entries containing the caret symbol (^) must be enclosed in quotes.
			The following sample code shows how you can use the projects parameter to purge old asset versions from an individual visual test named *Login*:
			`-projects Common.VisualTest.Login`
			The following sample code shows how you can use the projects parameter to purge old asset versions from all active data assets in the project *MyProject*:
			`-projects MyProject.ActiveData`
			When specifying asset types or individual assets with the `projects` parameter, you cannot specify the assettypes parameter. However, you can use the two parameters together when specifying projects. For example:
			`-projects Common -assettypes script`

Purge by Count	`pc` `purgebycount`	`-pc 10` `/purgebycount`	Deletes older asset versions in excess of the number specified. Either this parameter or the *Purge by Date* parameter is required. The use of both parameters is not permitted. The minimum number of versions you can keep is 1 and the maximum number of versions is 32767. You cannot delete the active version of an asset. This parameter accepts a single numerical value. For example, if you specify `10`, then the latest ten versions of an asset are retained and any older versions are deleted. **Note:** All actual checks are kept during purge and are deleted when results are deleted. Expected checks that are in use by results are also not deleted. After a purge, more expected checks that specified might be retained. For example, if 5 versions of an expected check are referenced by test results and you want to leave only 3 versions for each asset after the purge, all 5 versions of the check that are referenced by results are retained.

Parameter	Syntax	Example	Description
Purge by Date	`pd` `purgebydate`	`-pd 2011-03-28` `/purgebydate 2012-03-28`	Deletes asset versions modified before the specified date. Either this parameter or the *Purge by Count* parameter is required. The use of both parameters is not permitted. This parameter accepts a date between 1970-01-01 and 3000-12-31 in the following date format: `[YYYY-MM-DD]`.  **Note:** All actual checks are kept during purge and are deleted when results are deleted. Expected checks that are in use by results are also not deleted. This might lead to situations where expected checks are retained which are older than the date you have specified.
Info Mode	`i` `info`	`-i silent` `/info verbose`	Sets the type of information to display during the execution of the program. This parameter is optional. If omitted, the default mode of "Normal" is used. This parameter accepts the following values (syntax is indicated in brackets): **Normal [n|normal]** Outputs all errors and general information. **Silent [s|silent]** Outputs only errors that stop execution. **Verbose [v|verbose]** Outputs all warnings, errors, and general information.
Prompt	`o` `prompt`	`-o /prompt`	Displays a confirmation message box after displaying the asset purge report. This parameter is optional.

Help	`h`
	`?`
	`help`
		`-h /help`	Displays the help content and ignores all other parameters except for *Color Mode*.

Example 1

This example shows how to purge all results in the *Common* project, leaving five versions of each result:

```
stwpurge /dsn SilkTest /username admin /password admin /
purgebycount 5 /projects Common /assettypes result
```

Example 2

This example shows how to purge all versions of the visual test *LoginTest* that were created before December 31, 2016:

```
stwpurge /dsn SilkTest /username admin /password admin /
purgebydate 2016-12-31 /projects Common.VisualTest.LoginTest
```

Creating an Asset Using the Asset Browser

Most assets are created by inserting them in an existing visual test or script, or creating them during recording. However, you can create assets using the **Asset Browser**.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. Select the appropriate asset type in the **Asset Types** list to display the list of assets.
3. In the asset list, right-click the asset to create, and choose **New** > **Asset Type** to create a new asset of this type.

   To view a list of all asset types, choose **New**.

   All assets must follow asset naming conventions.

   New assets are added to the currently active project and are available only to other assets in that project or to assets in a project that references the active project. Change an active project by selecting it from the **Active** list in the **Projects** area of the window. Use the Common project to store assets used in multiple projects. Assets in the Common project are available to all projects that reference the Common project.

Opening an Asset from a Script

When you are editing a script, you can open an asset by right clicking it and selecting **Open Asset.** This will open the asset in the GUI.

If the asset is a reference to a file on the system, for example, referenced by `ImageClickFile`, the file will be opened by your system's default editor.

Opening an Asset

Open assets using the **Asset Browser**.

1. Choose **File** > **Open** or select the **Get Started** pane in the **Start Screen** and click **Asset Browser** in the **Tasks** area.
2. Select the appropriate asset type in the **Asset Types** list to display the list of assets.
3. Double-click the asset you want to open in the right pane.

Duplicating an Asset

A duplicated asset retains all the attributes of the original asset. Duplicating assets saves time when multiple assets of the same type are required that have similar attributes. After you have created and saved an asset, you can duplicate it at any time.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. Select the appropriate asset type in the **Asset Types** list to display the list of assets.
3. In the right pane of the **Asset Browser**, right-click the asset you want to duplicate and choose **Duplicate**. The **Duplicate Asset** dialog box opens.
4. Type a name of the duplicate asset and optionally change the project where the asset will reside and type a description for the new asset.
5. Click **OK**.

Renaming an Asset

After you create and save an asset, you can rename it at any time.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. Select the appropriate asset type in the **Asset Types** list to display the list of assets.
3. In the right pane of the **Asset Browser**, right-click the asset you want to rename and choose **Rename**.

   You can also select the asset that you want to rename and press **F2**.

   The **Rename Asset** dialog box opens.
4. Type a new name for the asset. and type a description for the asset.
5. *Optional:* Change the project where the asset resides to move the asset to another project.

   For example, use this when trying to move visual tests or VB .NET scripts to another project.
6. *Optional:* Type a description for the asset.
7. Click **OK**.
8. If the asset is referenced by other assets, specify what to do:

   - To rename the asset and to update the references, click **Rename this asset and update references**.
   - To rename the asset without updating the references, click **Rename this asset only**.

     Note: The referencing assets might not playback correctly.
   - To cancel renaming the asset, click **Cancel**.

   If you are moving the asset to a different project, and the new project is not referenced by the projects of a referencing asset, an error icon is shown for the referencing asset. Hover the mouse cursor over the error icon to see information about the error.

Deleting an Asset

After creating and saving an asset, you can delete it at any time.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. Under **Asset Types**, select the appropriate asset type.
3. In the list of assets, right-click the asset to remove and choose **Delete**. The **Delete** dialog box opens. If there are multiple versions of the asset, Silk Test Workbench gives you the option to delete the latest version or all versions.
4. If there are multiple versions of the asset and you want to delete them all, click **Delete all versions** and click **Yes**. Otherwise, click **Yes** to delete the current or only version of the asset or **No** to close the **Delete** dialog box without deleting the asset.

Filtering the List of Assets

Use the **Filters** list to display only the assets that meet specific criteria.

1. Click **Filter** in the **Asset Browser** window. The **Filter** dialog box opens.
2. Specify the filter or filters to use.

   If you specify more than one filter, an asset must meet all filter criteria to display. The following filters are included:

Created By	Select the name of a user to see all assets created by that user.
Creation Date	Specify the starting and ending dates to see all assets created during that period.
Last Modified By	Select the name of a user to see all assets modified by that user.
Last Modified Date	Specify the starting and ending dates to see all assets modified during that period.

3. To display only the assets that have a specific name in the **Asset Browser**, type the name into the **Name Filter** text box.

   The **Asset Browser** displays the assets that match the **Name Filter** and any additional filter criteria.
4. Click **OK**. Silk Test Workbench applies the filter to the list of asset items currently displayed.

   Note: To clear the filter and display all the items again, click **Filter** and then click **Default** on the **Filter** dialog box.

To display only the assets that have a specific name in the **Asset Browser**, type the name into the **Name Filter** text box. Any additional filtering is honored in the displayed asset set.

Searching for Assets

Use the **Name Filter** field of the **Asset Browser** to search for assets with a specific name.

1. Open the **Asset Browser**.
2. Select the asset type.
3. Click into the **Name Filter** field.

   You can also press **F3**.
4. Specify the name of the asset that you want to find.

   While you type the name of the asset, Silk Test Workbench displays only those assets that match the characters in the **Name Filter** field.

Sorting Assets

You can sort items in the **Asset Browser** according to the contents of any column. By default, the list of assets is sorted in ascending order.

1. Click the header of the column by which you want to sort the items displayed in the **Asset Browser**. For example, to sort items by date, click the **Creation Date** column header.
2. To reverse the sort direction, click the header of the column by which you previously sorted the items displayed in the **Asset Browser**.
   For example, to reverse the sort order of items by creation date, click the **Creation Date** column header a second time.

Setting the Default Behavior for Saving Assets

Set the default save behavior to either create a new version of an asset or replace the current version of the asset each time an asset is saved.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Global** in the **Options** menu tree and then click **Asset Management**. The **Asset Management** options display in the right side panel.

3. From the **Default save behavior** list box, select **Save as new version** or **Save as current version**. Silk Test Workbench creates a new version of an asset or replaces the current version of the asset each time an asset is saved from a toolbar button or shortcut key (`Ctrl+S`).
4. Click **OK**.

Setting the Maximum Number of Asset Versions

To limit the storage requirements of assets, including test results, you can specify a maximum number of versions of an asset to keep.

1. In the menu, select **Tools** > **Options**. The **Options** dialog box opens.
2. In the **Options** tree, expand **Global**.
3. Select **Asset Management**.
4. In the **Asset Management** page, type the number of versions that you want to keep into the **Maximum asset versions** text box.

   The default setting is 0, which means all versions of the asset are saved.
5. Click **OK**.

If the maximum number of versions is set and reached, and you save a new version of the asset, the oldest saved version of the asset is deleted. When importing an asset, all versions are imported, even if the number of versions exceeds the maximum version number. When saving a new version of such an imported asset, the oldest versions of the asset are deleted to match the specified maximum number of assets.

Note: When limiting the number of asset versions, old test results are automatically purged.

Viewing Multiple Versions of an Asset

Silk Test Workbench automatically stores versions of all asset types. Each time you modify an asset, a new version of that asset is stored in the database. At any time, you can view a list of all the versions of an asset and version information, including by whom the asset was last modified, the date and time of the modification, and the version number.

Silk Test Workbench only references the active version of an asset.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. Select the appropriate asset type in the **Asset Types** list to display the list of assets.
3. In the list of assets, right-click the desired asset and choose **Show All Versions**. The **Version Detail** dialog box opens.
4. You can use the buttons across the bottom of the **Version Detail** dialog box to open, duplicate, delete, and close versions of the asset.
5. Make any version the currently active version by selecting it and clicking **Make Active Version**.

Asset Naming Conventions

When you add an asset in the **New Asset** dialog box, Silk Test Workbench checks the name to verify compliance to established naming conventions.

Silk Test Workbench uses the following naming conventions:

- Asset names must be between 1 and 126 characters in length.

- Asset names cannot contain any of the following characters: \ / < > " ' : * ? | or =.
- An asset name cannot be the same as the name of another asset of the same type (or group) in the same project.

Importing and Exporting Assets

You can transfer assets from one database to another by using the import and export utilities. The import and export of assets is designed to simplify project management, share assets across a WAN, and utilize external version control systems. To import or export assets, you can either use the **Asset Import** wizard or the **Asset Export** wizard. For either method, you can customize the import or export of assets depending on your needs and preferences.

The assets eligible for import and export are project assets and system assets.

Project Assets	For detailed information, see *Asset Types*.
System Assets	Record and playback options.

Importing Assets

Import assets from one database to another by using the **Asset Import** wizard.

1. Choose **File** > **Import Assets**. The **Silk Test Workbench Asset Import** wizard opens.

   To bypass the **Welcome** page on subsequent uses of the wizard and all other Silk Test Workbench wizards, check the check box at the bottom of the page. Checking this check box updates the **Show Wizard Welcome Screens** option in **General** options.

   You can resize the wizard by dragging the lower-right corner of any page to the desired height and width.
2. Click **Next**. The **Import File Name** page opens.
3. Specify the ZIP file which contains the assets that you want to import.
4. Click **Next**. The **Items to Import** page opens.
5. Select the items to import by performing one of the following steps:

   - To import all assets, click **Import all assets**.

     To import all versions of the assets, uncheck the **Import only the latest version of each asset** check box.
   - To import specific assets, click **Import only the assets I select**.

     To import specific versions for each asset, check the **Show all versions of each asset** check box . Then, click **Next**.

     Check the check boxes for the assets that you want to import.
6. Click **Next**. The **Select Projects** page opens.
7. **Optional:** Select the target project from the **Target** column to specify into which project the assets from a source project should be imported.

   Note: If the current user is an administrator, there is an additional entry in the **Target** list to create a new project into which the assets should be imported.

   A project is not included in the **Target** list under the following circumstances:

   - The project in the import file does not exist in the database and the current user is not an administrator
   - The current user does not have at least Executor access to the existing project.

Note: If you are an administrator, you can select to create a new project into which the assets should be imported.

8. Click **Next**. The **Conflict Resolution** page opens.
9. Select the conflict resolution method for project assets in the event that the name of an imported asset conflicts with the name of an existing asset.

   Choices include:

   - **Append imported asset as latest version of existing asset** – This option creates a new version of the asset in the database. If there is more than one version of the asset being imported, the asset is imported with the lowest version first and highest version last. This means that the asset with the highest version number becomes the latest asset version in the database.
   - **Prompt me to append, rename, or skip the imported asset** – This is an interactive option that produces a dialog box for each conflict requesting you to either append, rename, or skip the imported asset.
   - **Do not import the asset** – Skips the import of the asset.
10. Select the conflict resolution method for system assets in the event that the name of an imported asset conflicts with the name of an existing asset.

    Choices include:

    - **Prompt me to rename, or skip the imported asset** – This is an interactive option that produces a dialog box for each conflict requesting you to either rename or skip the imported asset.
    - **Do not import the asset** – Skips the import of the asset.
11. Click **Next**. The **Start Importing Assets** page opens.
12. Review the summary.

    While the assets are being imported:

    - All projects that are listed in the ZIP file and that do not exist in the target database are created.
    - No additional project references are added for projects that already exist in the database. The summary shows such conflicts so that you can manually add references between the projects, if required.

      Note: When importing a ZIP file from a Silk Test Workbench version prior to 19.0, the references between projects are not included in the ZIP file and Silk Test Workbench adds a reference to the Common project to all newly created projects.
13. Click **Next** to begin the import. The **Import Progress** window displays a progress bar that shows the level of completion. The **Import Wizard Complete** page opens after the import finishes.
14. *Optional*: Click **Save** to save the detail report.
15. Click **Finish**.

Note: When using Silk Test Workbench 18.5 or later, you cannot import assets that have been exported with a Silk Test Workbench version prior to 18.5. To import such exported assets, use the **StwConvertExportedXML** program from the command line to convert the assets to the required format. This program has the following arguments:

```
STWConvertExportedXML [-overwrite] [-?] path_to_old_export
path_to_new_export.zip
```

- `path_to_old_export` is the path to the file or folder that was generated by the old version of Silk Test Workbench.
- `path_to_new_export.zip` is the path to the ZIP file into which the converted export files should be saved.
- If `-overwrite` is specified, and the ZIP file exists, the program overwrites the file. If `-overwrite` is not specified, and the ZIP file exists, an error message is printed and the program exits.
- `-?` causes help text to be shown and exits the program.

Note: When you import an asset, all historical data is replaced with information about the current import. For example, the Creator, Creation Date, Last Modified By, and Modified Dates reflect the user that performed the import and the date and times of the import.

Exporting Assets

Export assets from one database to another by using the **Asset Export** wizard.

Note: Never modify an exported file. Doing so could potentially corrupt your Silk Test Workbench database upon import of the file. If an export occurs while an asset is being edited, the edited asset is not exported. Therefore, make sure to complete and save any edits to an asset before performing an export.

1. Choose **File** > **Export Assets**. The Silk Test Workbench **Asset Export Wizard** opens.

   To bypass the **Welcome** page on subsequent uses of the wizard and all other Silk Test Workbench wizards, check the check box at the bottom of the page. Checking this check box updates the **Show Wizard Welcome Screens** option in **General** options.

   You can resize the wizard by dragging the lower-right corner of any page to the desired height and width.
2. Click **Next**. The **What to Export** page opens.
3. To export all assets, perform the following steps:
   a) Click **Export all assets**.
   b) Uncheck the **Export only the latest version of each asset** check box to export every version of each asset, or check the check box to include only the latest version in the export.
4. To export specific assets, perform the following steps:
   a) Click **Export only the assets I select**.
   b) To export specific versions for each asset, check the **Show all versions of each asset** check box.

      Note: If you choose to export more than one version of an asset, note that the version information may change on import, depending on the versions that you select. For example, if you choose to export version 1 and version 5 of an asset, during the import the versions numbers are updated to version 1 and version 2.
   c) Check the **Export assets associated with Visual tests and .NET Scripts** check box to export every associated assets, including all object maps from all referenced projects.
   d) Click **Next**. The **Items to Export** page opens.
   e) In the left pane, expand the folders that contain the items that you want to export and check the check boxes for the assets that you want to export.

      A blue check mark indicates that the item selected and all its subitems are selected for export. A gray check mark indicates that only a partial selection of subitems has occurred.
5. Click **Next**. The **Where to store the data** page opens.
6. Locate or create the ZIP file to which you want to export the assets.

   All exported assets are bundled into the specified file.
7. Click **Next**. The **Start Exporting Assets** page opens. This page provides a summary of the assets for export.
8. Review the summary and then click **Next** to begin the export. The **Export Progress** window displays a progress bar that shows the level of completion. The **Export Wizard Complete** page opens after the export finishes.
9. *Optional*: Click **Save** to save the detail report.
10. Click **Finish**.

Asset Import and Export Permissions

Asset import functionality can be limited or disabled depending on the permissions assigned to a user. Permissions can be assigned uniquely for each project. For example, a user with read-only permission for

ProjectA can be assigned full access permission to *ProjectB*. As a result, import functionality for this user is disabled for *ProjectA* but enabled for *ProjectB*.

Asset export functionality is generally not restricted. Any user with read-only access or better can export both project and system assets.

The following table shows the capability of each user role to import project assets and system assets, which are defined as:

- Project Assets – For detailed information, see *Asset Types*.
- System Assets – Record and playback options.

Asset Import Permissions

Only an administrator or a user with full access can import all project and system assets. For a complete listing of import permissions, refer to the following table:

User Role	Import Project Assets?	Import System Assets?
No Access	No	Yes
Read Only	No	Yes
Executor	Yes [1]	Yes
Script Writer	Yes	Yes
Full Access	Yes	Yes

[1] – Only the import of results is permitted.

Asset Export Permissions

User roles with read-only access or better can export all project and system assets.

Object Recognition

Silk Test Workbench identifies any control in the application under test (AUT) by combining the name of the control class and a collection of prioritized attributes into a unique XPath locator. If the combined XPath locator does not uniquely identify the control, Silk Test Workbench additionally adds an index to the locator. During recording, Silk Test Workbench allows you to select an alternative locator for a control from the list in the **Locator** field of the **Choose Action** dialog.

Within Silk Test Workbench, literal references to identified objects are referred to as *locators*. Silk Test Workbench uses locators to find and identify objects in the application under test (AUT). Locators are a subset of the XPath query language, which is a common XML-based language defined by the World Wide Web Consortium, W3C.

Silk Test Workbench automatically records locators when you record a visual test or script. By default, Silk Test Workbench substitutes an alias or logical name, called an *object map item*, instead of using the literal reference to the control or window's locator name. Visual tests and scripts reference objects based on the object map item rather than the actual locator name.

If you turn off object maps, locators are included in the visual test or script context instead of object map items.

Locator Basic Concepts

Silk Test Workbench supports a subset of the XPath query language. For additional information about XPath, see *http://www.w3.org/TR/xpath20/*.

XPath expressions rely on the current context, the position of the object in the hierarchy on which the `Find` method was invoked. All XPath expressions depend on this position, much like a file system. For example:

- "//Shell" finds all shells in any hierarchy starting from the current context.
- "Shell" finds all shells that are direct children of the current context.

Additionally, some XPath expressions are context sensitive. For example, myWindow.find(xPath) makes myWindow the current context.

Dynamic object recognition uses a `Find` or `FindAll` functions to identify an object in a test case.

Object Type and Search Scope

A locator typically contains the type of object to identify and a search scope. The search scope is one of the following:

- //
- /

Locators rely on the current object, which is the object for which the locator is specified. The current object is located in the object hierarchy of the application's UI. All locators depend on the position of the current object in this hierarchy, much like a file system.

XPath expressions rely on the *current context*, which is the position of the object in the hierarchy on which the `Find` method was invoked. All XPath expressions depend on this position, much like a file system.

Note:

The object type in a locator for an HTML element is either the HTML tag name or the class name that Silk Test Workbench uses for this object. For example, the locators `//a` and `//DomLink`, where

`DomLink` is the name for hyperlinks in Silk Test Workbench, are equivalent. For all non-HTML based technologies only the Silk Test Workbench class name can be used.

Example

- `//a` identifies hyperlink objects in any hierarchy relative to the current object.
- `/a` identifies hyperlink objects that are direct children of the current object.

Note: <*a*> is the HTML tag for hyperlinks on a Web page.

Example

The following code sample identifies the first hyperlink in a browser. This example assumes that a variable with the name *browserWindow* exists in the script that refers to a running browser instance. Here the type is "a" and the current object is *browserWindow*.

```
Dim link As DomLink = browserWindow.DomLink("//a")
```

Using Attributes to Identify an Object

To identify an object based on its properties, you can use locator attributes. The locator attributes are specified in square brackets after the type of the object.

Example

The following sample uses the `textContents` attribute to identify a hyperlink with the text *Home*. If there are multiple hyperlinks with the same text, the locator identifies the first one.

```
 Dim link as DomLink = browserWindow.DomLink("//
a[@textContents='Home']")
```

Locator Syntax

Silk Test Workbench supports a subset of the XPath query language to locate UI controls.

The following table lists the constructs that Silk Test Workbench supports.

Note: <*a*> is the HTML tag for hyperlinks on a Web page.

The following table lists the locator constructs that Silk Test Workbench does not support.

Using Locators in Scripts

Within Silk Test Workbench, literal references to identified objects are referred to as *locators*. For convenience, you can use shortened forms for the locator strings in scripts. Silk Test Workbench automatically expands the syntax to use full locator strings when you playback a script. When you manually code a script, you can omit the following parts in the following order:

- The search scope, `//`.
- The object type name. Silk Test Workbench defaults to the class name.

- The surrounding square brackets of the attributes, `[ ]`.

When you manually code a script, we recommend that you use the shortest form available.

Note: You can use shortened forms for the locator strings only when you use a .NET script. For visual tests, you must use full locator strings. When you identify an object, the full locator string is captured by default.

The following locators are equivalent:

- The first example uses the full locator string.

  ```
  _desktop.DomLink("//BrowserApplication//BrowserWindow//
  a[@textContents='Home']").Select()
  ```

  To confirm the full locator string, use the **Identify Object** dialog box.
- The second example works when the browser window already exists.

  ```
  browserWindow.DomLink("//a[@textContents='Home']").Select()
  ```

  Alternatively, you can use the shortened form.

  ```
  browserWindow.DomLink("@textContents='Home'").Select()
  ```

To find an object that has no real attributes for identification, use the index. For instance, to select the second hyperlink on a Web page, you can type:

```
browserWindow.DomLink("[2]").Select()
```

Additionally, to find the first object of its kind, which might be useful if the object has no real attributes, you can type:

```
browserWindow.DomLink().Select()
```

Using the Find Method

Instead of using methods like `.DomLink`, you can use the `Find` method to identify a single object with a locator.

Note: The `Find` method can only use full locators, the shortened locator form is not supported.

Instead of typing:

```
_desktop.DomLink("//a[@textContents='Home']").Select()
```

you can type:

```
_desktop.Find(Of DomLink)("//a[@textContents='Home']").Select()
```

The `.DomLink` method also uses the `Find` method internally. Prefer using the `.DomLink` method, because it is more concise than the `Find` method.

Using Locators to Check if an Object Exists

You can use the `Exists` method to determine if an object exists in the application under test.

The following code checks if a hyperlink with the text *Log out* exists on a Web page:

```
If (browserWindow.Exists( "//a[@textContents='Log out']" )) Then
  ' do something
End If
```

Using the Find method

You can use the `Find` method and the `FindOptions` method to check if an object, which you want to use later, exists.

The following code searches for a window and closes the window if the window is found:

```
Dim mainWindow As Window
mainWindow = _desktop.Find("//Window[@caption='My Window']", New
FindOptions(False))
If (mainWindow IsNot Nothing) Then
   mainWindow.CloseSynchron()
End If
```

Identifying Multiple Objects with One Locator

You can use the `FindAll` method to identify all objects that match a locator rather that only identifying the first object that matches the locator.

Example

The following code example uses the `FindAll` method to retrieve all hyperlinks of a Web page:

```
Dim links As IList(Of DomLink) = browserWindow.FindAll(Of
DomLink)("//a")
```

Locator Customization

This section describes how you can create stable locators that enable Silk Test Workbench to reliably recognize the controls in your application under test (AUT).

Silk Test Workbench relies on the identifiers that the AUT exposes for its UI controls and is very flexible and powerful in regards to identifying UI controls. Silk Test Workbench can use any declared properties for any UI control class and can also create locators by using the hierarchy of UI controls. From the hierarchy, Silk Test Workbench chooses the most appropriate items and properties to identify each UI control.

Silk Test Workbench can exclude dynamic numbers of controls along the UI control hierarchy, which makes the object recognition in Silk Test Workbench very robust against changes in the AUT. Intermediate grouping controls that change the hierarchy of the UI control tree, like formatting elements in Web pages, can be excluded from the object recognition.

Some UI controls do not expose meaningful properties, based on which they can be identified uniquely. Applications which include such controls are described as applications with *bad testability*. Hierarchies, and especially dynamic hierarchies, provide a good means to create unique locators for such applications. Applications with *good testability* should always provide a simple mechanism to identify UI controls uniquely.

One of the simplest and most effective practices to make your AUT easier to test is to introduce stable identifiers for controls and to expose these stable identifiers through the existing interfaces of the application.

Stable Identifiers

A *stable identifier* for a UI control is an identifier that does not change between invocations of the control and between different versions of the application, in which the UI control exists. A stable identifier needs to be unique in the context of its usage, meaning that no other control with the same identifier is accessible at the same time. This does not necessarily mean that you need to use GUID-style identifiers that are unique

in a global context. Identifiers for controls should be readable and provide meaningful names. Naming conventions for these identifiers will make it much easier to associate the identifier to the actual control.

Example: Is the caption a good identifier for a control?

Very often test tools are using the *caption* as the default identifier for UI controls. The caption is the text in the UI that is associated with the control. However, using the caption to identify a UI control has the following drawbacks:

- The caption is not stable. Captions can change frequently during the development process. For example, the UI of the AUT might be reviewed at the end of the development process. This prevents introducing UI testing early in the development process because the UI is not stable.
- The caption is not unique. For example, an application might include multiple buttons with the caption **OK**.
- Many controls are not exposing a caption, so you need to use another property for identification.
- Using captions for testing localized applications is cumbersome, as you need to maintain a caption for a control in each language and you also have to maintain a complex script logic where you dynamically can assign the appropriate caption for each language.

Creating Stable Locators

One of the main advantages of Silk Test Workbench is the flexible and powerful object-recognition mechanism. By using XPath notation to locate UI controls, Silk Test Workbench can reliably identify UI controls that do not have any suitable attributes, as long as there are UI elements near the element of interest that have suitable attributes. The XPath locators in Silk Test Workbench can use the entire UI control hierarchy or parts of it for identifying UI controls. Especially modern AJAX toolkits, which dynamically generate very complex Document Object Models (DOMs), do not provide suitable control attributes that can be used for locating UI controls.

In such a case, test tools that do not provide intelligent object-recognition mechanisms often need to use index-based recognition techniques to identity UI controls. For example, identify the n-th control with icon *Expand*. This often results in test scripts that are hard to maintain, as even minor changes in the application can break the test script.

A good strategy to create stable locators for UI controls that do not provide useful attributes is to look for an anchor element with a stable locator somewhere in the hierarchy. From that anchor element you can then work your way to the element for which you want to create the locator.

Silk Test Workbench uses this strategy when creating locators, however there might be situations in which you have to manually create a stable locator for a control.

Example: Locating Siblings of a Control

This topic describes how you can locate a control, which does not provide any meaningful attributes that can be used in locators, when a stable locator for a sibling of the control is available.

Assume that you have already identified the control **Item 0.0**, which has the following stable locator:

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item 0.0']
```

If you know that **Item 0.0** has a following-sibling of the type *a*, you can use the following code to build a stable locator for the sibling:

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item 0.0']/following-
sibling::a
```

You can also use the sibling approach to identify text fields. Text fields often do not provide any meaningful attributes that can be used in locators. By using the label of a text field, you could create a meaningful

locator for the text field, because the label is the best identifier for the text field from the perspective of a tester. You can easily use the label as a part of the locator for a test field by using the sibling approach. For example, if the text field is a preceding-sibling of a label with the text **User Name**, you can use the following locator:

```
/BrowserApplication//BrowserWindow//DIV[@textContent='User Name']/preceding-sibling::input[@type='text']
```

Example: Locating the Expand Icon in a Dynamic GWT Tree

The Google Widget Toolkit (GWT) is a very popular and powerful toolkit, which is hard to test. The dynamic tree control is a very commonly used UI control in GWT. To expand the tree, we need to identify the **Expand** icon element.

You can find a sample dynamic GWT tree at *http://samples.gwtproject.org/samples/Showcase/Showcase.html#!CwTree*.

The default locator generated by Silk Test Workbench is the following:

```
/BrowserApplication//BrowserWindow//DIV[@id='gwt-debug-cwTree-dynamicTree-root-child0']/DIV/DIV[1]//IMG[@border='0']
```

For the following reasons, this default locator is no reliable locator for identifying the **Expand** icon for the control **Item 0.0**:

- The locator is complex and built on multiple hierarchies. A small change in the DOM structure, which is dynamic with AJAX, can break the locator.
- The locator contains an index for some of the controls along the hierarchy. Index based locators are generally weak as they find controls by their occurrence, for example finding the sixth expand icon in a tree does not define the control well. An exception to that rule would be if the index is used to express different data sets that you want to identify, for example the sixth data row in a grid.

Often a good strategy for finding better locators is to search for siblings of elements that you need to locate. If you find siblings with better locators, XPath allows you to construct the locator by identifying those siblings. In this case, the tree item **Item 0.0** provides a better locator than the **Expand** icon. The locator of the tree item **Item 0.0** is a stable and simple locator as it uses the `@textContents` property of the control.

By default, Silk Test Workbench uses the property `@id`, but in GWT the `@id` is often not a stable property, because it contains a value like `='gwt-uid-<nnn>'`, where `<nnn>` changes frequently, even for the same element between different calls.

You can manually change the locator to use the `@textContents` property instead of the `@id`.

Original Locator:

```
/BrowserApplication//BrowserWindow//DIV[@id='gwt-uid-109']
```

Alternate Locator:

```
/BrowserApplication//BrowserWindow//DIV[@textContents='Item 0.0']
```

Or you can instruct Silk Test Workbench to avoid using `@id='gwt-uid-<nnn>'`. In this case Silk Test Workbench will automatically record the stable locator. You can do this by adding the text pattern that is used in `@id` properties to the locator attribute value blacklist. In this case, add `gwt-uid*` to the blacklist.

When inspecting the hierarchy of elements, you can see that the control **Item 0.0** and the **Expand** icon control have a joint root node, which is a `DomTableRow` control.

To build a stable locator for the **Expand** icon, you first need to locate **Item 0.0** with the following locator:

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item 0.0']
```

Then you need to go up two levels in the element hierarchy to the `DomTableRow` element. You express this with XPath by adding `/../..` to the locator. Finally you need to search from `DomTableRow` for the

Expand icon. This is easy as the **Expand** icon is the only IMG control in the sub-tree. You express this with XPath by adding `//IMG` to the locator. The final stable locator for the **Expand** icon looks like the following:

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item 0.0']/../..//IMG
```

Or even better, use the XPath ancestor axis to locate the **Expand** icon:

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item 0.0']/
ancestor::tr//IMG
```

Custom Attributes

Many UI technologies provide a mechanism that allows them to extend the set of predefined attributes of UI controls with custom attributes. These custom attributes can be used by the application developer to introduce stable identifiers that uniquely identify the control. Silk Test Workbench can access custom attributes of UI controls and can also use these custom attributes to identify UI controls.

Using special automation for the identification of UI controls has several advantages compared to using the defined attributes like `caption`. Being able to establish stable identifiers in the application code and to expose these identifiers through either custom attributes or defined automation properties leads to understandable and maintainable test-automation scripts, allowing you to start with your test automation early in the development process.

You can configure the attributes used for identification by using the flexible locator strategy of Silk Test Workbench.

Custom Attributes for Apache Flex Applications

Apache Flex applications use the predefined property `automationName` to specify a stable identifier for the Apache Flex control as follows:

```
<?xml version="1.0" encoding="utf-8"?>
  <s:Group xmlns:fx="http://ns.adobe.com/mxml/2009"
      xmlns:s="library://ns.adobe.com/flex/spark"
      xmlns:mx="library://ns.adobe.com/flex/mx" width="400" height="300">
    <fx:Script>
    …
    </fx:Script>
    <s:Button x="247" y="81" label="Button" id="button1" enabled="true"
click="button1_clickHandler(event)"
        automationName="AID_buttonRepeat"/>
    <s:Label x="128" y="123" width="315" height="18" id="label1"
verticalAlign="middle"
        text="awaiting your click" textAlign="center"/>
  </s:Group>
```

Apache Flex application locators look like the following:

```
…//SparkApplication//SparkButton[@caption='AID_buttonRepeat'
```

Attention: For Apache Flex applications, the *automationName* is always mapped to the locator attribute `caption` in Silk Test Workbench. If the *automationName* attribute is not specified, Silk Test Workbench maps the property `ID` to the locator attribute `caption`.

Java SWT Custom Attributes

You can add custom attributes to a test application to make a test more stable. For example, in Java SWT, the developer implementing the GUI can define an attribute (for example, `'silkTestAutomationId'`) for a widget that uniquely identifies the widget in the application. A tester using Silk Test Workbench can then add that attribute to the list of custom attributes (in this case, `'silkTestAutomationId'`), and can identify controls by that unique ID. Using a custom attribute is more reliable than other attributes like caption or index, since a caption will change when you translate the application into another language, and the index will change whenever another widget is added before the one you have defined already.

If more than one object is assigned the same custom attribute value, all the objects with that value will return when you call the custom attribute. For example, if you assign the unique ID, `'loginName'` to two different text fields, both fields will return when you call the `'loginName'` attribute.

Java SWT Example

If you create a button in the application that you want to test using the following code:

```
Button myButton = Button(parent, SWT.NONE);

myButton.setData("SilkTestAutomationId", "myButtonId");
```

To add the attribute to your XPath query string in your test, you can use the following query:

```
Dim button =
desktop.PushButton("@SilkTestAutomationId='myButton'")
```

To enable a Java SWT application for testing custom attributes, the developers must include custom attributes in the application. Include the attributes using the `org.swt.widgets.Widget.setData(String key, Object value)` method.

Custom Attributes for Web Applications

HTML defines a common attribute `ID` that can represent a stable identifier. By definition, the ID uniquely identifies an element within a document. Only one element with a specific ID can exist in a document.

However, in many cases, and especially with AJAX applications, the ID is used to dynamically identify the associated server handler for the HTML element, meaning that the ID changes with each creation of the Web document. In such a case the ID is not a stable identifier and is not suitable to identify UI controls in a Web application.

A better alternative for Web applications is to introduce a new custom HTML attribute that is exclusively used to expose UI control information to Silk Test Workbench.

Custom HTML attributes are ignored by browsers and by that do not change the behavior of the AUT. They are accessible through the DOM of the browser. Silk Test Workbench allows you to configure the attribute that you want to use as the default attribute for identification, even if the attribute is a custom attribute of the control class. To set the custom attribute as the default identification attribute for a specific technology domain, click **Tools** > **Options** > **Record** and select the technology domain.

The application developer just needs to add the additional HTML attribute to the Web element.

Original HTML code:

```
<A HREF="http://abc.com/control=4543772788784322..." <IMG
src="http://abc.com/xxx.gif" width=16 height=16> </A>
```

HTML code with the new custom HTML attribute *AUTOMATION_ID*:

```
<A HREF="http://abc.com/control=4543772788784322..."
AUTOMATION_ID = "AID_Login" <IMG src="http://abc.com/xxx.gif"
width=16 height=16> </A>
```

When configuring the custom attributes, Silk Test Workbench uses the custom attribute to construct a unique locator whenever possible. Web locators look like the following:

```
…//DomLink[@AUTOMATION_ID='AID_Login'
```

Example: Changing ID

One example of a changing ID is the Google Widget Toolkit (GWT), where the ID often holds a dynamic value which changes with every creation of the Web document:

```
ID = 'gwt-uid-<nnn>'
```

In this case <nnn> changes frequently.

Custom Attributes for Windows Forms Applications

Windows Forms applications use the predefined automation property `automationId` to specify a stable identifier for the Windows forms control.

Silk Test Workbench automatically will use this property for identification in the locator. Windows Forms application locators look like the following:

```
/FormsWindow//PushButton[@automationId='btnBasicControls']
```

Custom Attributes for WPF Applications

WPF applications use the predefined automation property `AutomationProperties.AutomationId` to specify a stable identifier for the WPF control as follows:

```
<Window x:Class="Test.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
  <Grid>
    <Button AutomationProperties.AutomationId="AID_buttonA">The
Button</Button>
  </Grid>
</Window>
```

Silk Test Workbench automatically uses this property for identification in the locator. WPF application locators look like the following:

```
/WPFWindow[@caption='MainWindow']//WPFButton[@automationId='AID_buttonA']
```

Troubleshooting Performance Issues for XPath

When testing applications with a complex object structure, for example complex web applications, you may encounter performance issues, or issues related to the reliability of your scripts. This topic describes how you can improve the performance of your scripts by using different locators than the ones that Silk Test Workbench has automatically generated during recording.

Note: In general, we do not recommend using complex locators. Using complex locators might lead to a loss of reliability for your tests. Small changes in the structure of the tested application can break such a complex locator. Nevertheless, when the performance of your scripts is not satisfying, using more specific locators might result in tests with better performance.

The following is a sample element tree for the application MyApplication:

```
Root
  Node id=1
    Leaf id=2
    Leaf id=3
    Leaf id=4
    Leaf id=5
  Node id=6
    Node id=7
      Leaf id=8
      Leaf id=9
    Node id=9
      Leaf id=10
```

You can use one or more of the following optimizations to improve the performance of your scripts:

- If you want to locate an element in a complex object structure , search for the element in a specific part of the object structure, not in the entire object structure. For example, to find the element with the

identifier 4 in the sample tree, if you have a query like `Root.Find("//Leaf[@id='4']")`, replace it with a query like `Root.Find("/Node[@id='1']/Leaf[@id='4']")`. The first query searches the entire element tree of the application for leafs with the identifier 4. The first leaf found is then returned. The second query searches only the first level nodes, which are the node with the identifier 1 and the node with the identifier 6, for the node with the identifier 1, and then searches in the subtree of the node with the identifier 1 for all leafs with the identifier 4.

- When you want to locate multiple items in the same hierarchy, first locate the hierarchy, and then locate the items in a loop. If you have a query like `Root.FindAll("/Node[@id='1']/Leaf")`, replace it with a loop like the following:

```
Public Sub Main()
  Dim node As TestObject

  node = _desktop.Find("//Node[@id='1']")
  For i As Integer = 1 To 4 Step 1
    node.Find("/Leaf[@id='"+i+"']")
  Next

End Sub
```

Silk Test Open Agent

The Silk Test Open Agent is the software process that translates the commands in your scripts into GUI-specific commands. In other words, the Open Agent drives and monitors the application that you are testing.

One Agent can run locally on the host machine. In a networked environment, any number of Agents can replay tests on remote machines. However, you can record only on a local machine.

Starting the Silk Test Open Agent

Before you can create a test or run a sample script, the Silk Test Open Agent must be running. Typically, the Agent starts when you launch the product. If you must manually start the Open Agent, perform this step.

Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Tools** > **Silk Test Open Agent** or (in Microsoft Windows 10) **Start** > **Silk** > **Silk Test Open Agent**. The Silk Test Open Agent icon displays in the system tray.

Stopping the Open Agent After Test Execution

You can stop the Open Agent by using the command-line option -shutDown or from a script, to ensure that the agent does not continue running after the end of the test execution. The following code sample shows how you can stop the Open Agent from the command line:

```
openAgent.exe -shutDown
```

To stop the agent from a script:

1. Open or create a script that is executed when the test execution is finished.
   For example, open an existing script that is used for cleanup after test execution.
2. Add the `ShutDown` method to the script.

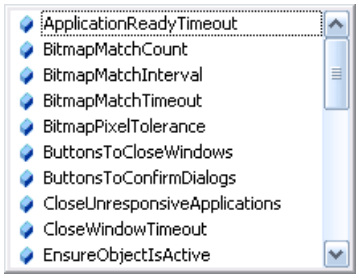
Note: The Open Agent will restart as soon as the agent is required by another script.

Agent Options

This section describes the options that can be manipulated with the `GetOption` and `SetOption` methods.

Note: By default, you can set global options by choosing **Tools** > **Options** in the menu.

To set or get agent options in a script, use the `GetOption` or `SetOption` methods. For instance, when you type, `Agent.SetOption(Options.`, the agent options display automatically in the active editor window with the auto-completion and syntactical assistance technology. For instance, you might see the following options in the active editor window:



An example for inserting an option into a script would be the following:

```
'VB .NET code
Agent.SetOption(Options.ApplicationReadyTimeout, 100000)
```

Alternatively, you can use the agent option name in scripts. For instance, you can type:

```
'VB .NET code
Agent.SetOption("OPT_APPREADY_TIMEOUT", 100000)
```

You can use the **Open Agent Settings** to add some custom options to visual tests. Navigate to **Tools** > **Options**, select **Record** or **Playback**, depending on what type of option you want to set, and add the new option as a name-value pair. You should only use this functionality in specific cases, and only if suggested by support.

.NET Option Name	Agent Option Name	Constant Type	Description
ApplicationReadyTimeout	OPT_APPREADY_TIMEOUT	NUMBER	Specifies the number of milliseconds to wait for a newly launched application to become ready. If the application is not ready within the specified timeout, Silk Test Workbench raises an exception.
BitmapMatchCount	OPT_BITMAP_MATCH_COUNT	INTEGER	Specifies the number of consecutive snapshots that must be the same for the bitmap to be considered stable. Snapshots are taken up to the number of seconds specified by OPT_BITMAP_MATCH_TIMEOUT, with a pause specified by OPT_BITMAP_MATCH_INTERVAL occurring between each snapshot. By default, this is 0.
BitmapMatchInterval	OPT_BITMAP_MATCH_INTERVAL	REAL	Specifies the time interval between snapshots to use for ensuring the stability of the bitmap image. The snapshots are taken up to the time specified by OPT_BITMAP_MATCH_TIMEOUT. By default, this is 0.1.
BitmapMatchTimeout	OPT_BITMAP_MATCH_TIMEOUT	REAL	Specifies the total time allowed for a bitmap image to become stable. During the time period, Silk Test Workbench takes multiple snapshots of the image, waiting the number of

.NET Option Name	Agent Option Name	Constant Type	Description
			seconds specified with OPT_BITMAP_MATCH_TIMEOUT between snapshots. If the value returned by OPT_BITMAP_MATCH_TIMEOUT is reached before the number of bitmaps specified by OPT_BITMAP_MATCH_COUNT match, Silk Test Workbench stops taking snapshots and raises the exception E_BITMAP_NOT_STABLE. By default, this is 5.
BitmapPixelTolerance	OPT_BITMAP_PIXEL_TOLERANCE	INTEGER	Specifies the number of pixels of difference below which two bitmaps are considered to match. If the number of pixels that are different is smaller than the number specified with this option, the bitmaps are considered identical. The maximum tolerance is 32767 pixels. By default, this is 0.
ButttonsToCloseWindows	OPT_CLOSE_WINDOW_BUTTONS	LIST OF STRING	Specifies the buttons used to close windows with the `CloseSynchron` method.
ButttonsToConfirmDialogs	OPT_CLOSE_CONFIRM_BUTTONS	LIST OF STRING	Specifies the buttons used to close confirmation dialog boxes that appear when closing windows with the `CloseSynchron` method.
CloseUnresponsiveApplications	OPT_KILL_HANGING_APPS	BOOLEAN	Specifies whether unresponsive applications are closed. An application is unresponsive if communication between the Agent and the application fails, e.g. times out. Set this option to TRUE when testing applications that cannot run multiple instances. By default, this is FALSE.
CloseWindowTimeout	OPT_CLOSE_WINDOW_TIMEOUT	NUMBER	Specifies the number of milliseconds to wait before the next close strategy is tried. The Agent executes four close attempts before failing, so the total time before a close fails is four times the value you specify.
Compatibility	OPT_COMPATIBILITY	STRING	Enables you to use the Silk Test Workbench behavior of the specified Silk Test Workbench version for specific features, when the behavior of these features has changed in the latest version.

.NET Option Name	Agent Option Name	Constant Type	Description
			Example strings: • `12` • `11.1` • `13.0.1` By default, this option is not set.
EnsureObjectIsActive	OPT_ENSURE_ACTIVE_OBJDEF	BOOLEAN	Ensures that the target object is active. By default, this is FALSE.
EnableAccessibility	OPT_ENABLE_ACCESSIBILITY	BOOLEAN	TRUE to enable Accessibility when you are testing a Win32 application and Silk Test Workbench cannot recognize objects. Accessibility is designed to enhance object recognition at the class level. FALSE to disable Accessibility. By default, this is FALSE.  **Note:** For Mozilla Firefox and Google Chrome, Accessibility is always activated and cannot be deactivated.
EnableMobileWebviewFallbackSupport	OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT	BOOLEAN	Enables mobile native fallback support for hybrid mobile applications that are not testable with the default browser support. By default, this is FALSE.
HangAppTimeOut	OPT_HANG_APP_TIME_OUT	NUMBER	Specifies the **Unresponsive application timeout**, which is the timeout for pending playback actions. The default value is 5000 milliseconds.
HighlightObjectDuringPlayback	OPT_REPLAY_HIGHLIGHT	BOOLEAN	Specifies whether the current object is highlighted during playback. By default, this is FALSE, which means that objects are not highlighted by default.
KeyboardEventDelay	OPT_KEYBOARD_INPUT_DELAY	NUMBER	Sets the delay in milliseconds between playback of keyboard strokes. Be aware that the optimal number you select can vary, depending on the application that you are testing. For example, if you are testing a Web application, a setting of 1 millisecond radically slows down the browser. However, setting this to 0 (zero) may cause basic application testing to fail.

.NET Option Name	Agent Option Name	Constant Type	Description
KeysToCloseDialogs	OPT_CLOSE_DIALOG_KEYS	LIST OF STRING	Specifies the keystroke sequence to close dialog boxes that open after trying to close a window with the `CloseSynchron` method. Examples include: <ESC>, <Alt+F4>.
LocatorAttributesCaseSensitive	OPT_LOCATOR_ATTRIBUTES_CASE_SENSITIVE	BOOLEAN	Set to `Yes` to add case-sensitivity to locator attribute names, or to `No` to match the locator names case insensitive. The names of locator attributes for mobile Web applications are always case insensitive, and this option is ignored when recording or replaying mobile Web applications.
MenuItemsToCloseWindows	OPT_CLOSE_WINDOW_MENUS	LIST OF STRING	Specifies the menu items used to close windows with the `CloseSynchron` method. Examples include: "File/Exit*", "File/Quit*".
MouseEventDelay	OPT_MOUSE_INPUT_DELAY	NUMBER	Specifies the delay in milliseconds used before each mouse event.
ObjectResolveRetryInterval	OPT_WAIT_RESOLVE_OBJDEF_RETRY	NUMBER	Specifies the time in milliseconds to wait before re-trying to find an object that could not be immediately resolved during playback. When the ObjectResolveTimeout runs out, no further retries are attempted.
ObjectResolveTimeout	OPT_WAIT_RESOLVE_OBJDEF	NUMBER	Specifies the time in milliseconds to wait for an object to be resolved during playback. As soon as the object is resolved, Silk Test Workbench can recognize it.
PlaybackMode	OPT_REPLAY_MODE	NUMBER	Defines how controls are replayed. Use low level to replay each control using the mouse and keyboard. Use high level to use the API to replay each control. All controls have a default playback mode assigned. When the default replay mode is selected, each control uses its default playback mode. The default mode delivers the most reliable results. Selecting low or high level playback overrides the playback mode of all controls with the playback mode selected. Possible values include 0, 1 and 2. 0 is default, 1 is high level, 2 is low level. By default, this is 0.

.NET Option Name	Agent Option Name	Constant Type	Description
PostReplayDelay	OPT_POST_REPLAY_DELAY	NUMBER	Specifies the time in milliseconds to wait after invoking a function or setting a property.
RemoveFocusOnCaptureText	OPT_REMOVE_FOCUS_ON_CAPTURE_TEXT	BOOLEAN	Set to `Yes` to take the focus off the application under test during a text capture. By default, this is set to `No`, leaving the focus on the application under test. A text capture is performed during recording and replay by the following methods: • `TextClick` • `TextCapture` • `TextExists` • `TextRect`
SuspensionTimeout	OPT_NTF_SUSPENSION_TIMEOUT	NUMBER	Specifies the maximum time in milliseconds to wait for an answer from the NTF. If the time is exceeded, the communication between NTF and the Open Agent is considered as suspended.
SyncTimeout	OPT_SYNC_TIMEOUT	NUMBER	Specifies the maximum time in milliseconds for an object to be ready. **Note:** When you upgrade from a Silk Test version prior to Silk Test 13.0, and you had set the OPT_XBROWSER_SYNC_TIMEOUT option, the **Options** dialog box will display the default value of the OPT_SYNC_TIMEOUT, although your timeout is still set to the value you have defined.
TransparentClasses	OPT_TRANSPARENT_CLASSES	LIST OF STRING	To simplify the object hierarchy and to shorten the length of the lines of code in your test scripts and functions, you can suppress the controls for certain unnecessary classes in the following technologies: • Win32. • Java AWT/Swing. • Java SWT/Eclipse. • Windows Presentation Foundation (WPF). Specify the names of any classes that you want to ignore during recording and playback.

.NET Option Name	Agent Option Name	Constant Type	Description
	OPT_WPF_CHECK_DISPATCHER_FOR_IDLE	BOOLEAN	For some WPF applications the Silk Test synchronization might not work due to how certain controls are implemented, resulting in Silk Test Workbench not recognising when the WPF application is idle. Setting this option to FALSE disables the WPF synchronization and prevents Silk Test Workbench from checking the WPF dispatcher, which is the thread that controls the WPF application. Set this option to FALSE to solve synchronization issues with certain WPF applications. By default, this is TRUE.
WPFCustomClasses	OPT_WPF_CUSTOM_CLASSES	LIST OF STRING	Specify the names of any WPF classes that you want to expose during recording and playback. For example, if a custom class called *MyGrid* derives from the WPF `Grid` class, the objects of the *MyGrid* custom class are not available for recording and playback. `Grid` objects are not available for recording and playback because the `Grid` class is not relevant for functional testing since it exists only for layout purposes. As a result, `Grid` objects are not exposed by default. In order to use custom classes that are based on classes that are not relevant to functional testing, add the custom class, in this case *MyGrid*, to the **OPT_WPF_CUSTOM_CLASSES** option. Then you can record, playback, find, verify properties, and perform any other supported actions for the specified classes.
WPFPrefillItems	OPT_WPF_PREFILL_ITEMS	BOOLEAN	Defines whether items in a `WPFItemsControl`, like `WPFComboBox` or `WPFListBox`, are pre-filled during recording and playback. WPF itself lazily loads items for certain controls, so these items are not available for Silk Test Workbench if they are not scrolled into view. Turn pre-filling on, which is the default setting, to additionally access items that are not accessible without scrolling them into view. However, some applications have problems when the items are pre-filled by Silk

.NET Option Name	Agent Option Name	Constant Type	Description
			Test Workbench in the background, and these applications can therefore crash. In this case turn pre-filling off.
XbrowserEnableIframeSupport	OPT_XBROWSER_ENABLE_IFRAME_SUPPORT	BOOLEAN	Specifies whether to enable iframe and frame support for browsers. If you are not interested in the content of the iframes in a web application, disabling the iframe support might improve replay performance. For example, disabling the iframe support might significantly improve replay performance for web pages with many adds and when testing in a mobile browser. This option is ignored by Internet Explorer. This option is enabled by default.
XBrowserExcludeIFrames	OPT_XBROWSER_EXCLUDE_IFRAMES	STRING	Every entry in the list defines an attribute name and the corresponding value. All iframes and frames that do not match at least one of the entries are considered during testing. Wildcards are allowed, for example the entry "src:*advertising*" would exclude <IFRAME src=http://my.domain/advertising-banner.html>. This option is ignored by Internet Explorer. If the list is empty, all iframes and frames are considered during testing. Separate multiple entries with a comma.
XBrowserFindHiddenInputFields	OPT_XBROWSER_FIND_HIDDEN_INPUT_FIELDS	BOOLEAN	Specifies whether to display hidden input fields, which are HTML fields for which the tag includes `type="hidden"`. The default value is TRUE.
XBrowserIncludeIFrames	OPT_XBROWSER_INCLUDE_IFRAMES	STRING	Every entry in the list defines an attribute name and the corresponding value. All iframes and frames that do not match at least one of the entries are excluded. Wildcards are allowed, for example the entry "name:*form" would include <IFRAME name="user-form" src=…>. This option is ignored by Internet Explorer. If the list is empty, all iframes and frames are considered during testing. Separate multiple entries with a comma.
XBrowserSynchronizationMode	OPT_XBROWSER_SYNC_MODE	STRING	Configures the supported synchronization mode for HTML or AJAX. Using the HTML mode ensures that all HTML documents are in an

.NET Option Name	Agent Option Name	Constant Type	Description
			interactive state. With this mode, you can test simple Web pages. If more complex scenarios with Java script are used, it might be necessary to manually script synchronization functions, such as `WaitForObject`, `WaitForProperty`, `WaitForDisappearance`, or `WaitForChildDisappearance`. Using the AJAX mode eliminates the need to manually script synchronization functions. By default, this value is set to **AJAX**.
XBrowserSynchronizationTimeout	OPT_XBROWSER_SYNC_TIMEOUT	NUMBER	Specifies the maximum time in milliseconds for an object to be ready.  **Note:** Deprecated. Use the option OPT_SYNC_TIMEOUT instead.

XBrowserSynchronizationURLExcludes	OPT_XBROWSER_SYNC_EXCLUDE_URLS	STRING	Specifies the URL for the service or Web page that you want to exclude during page synchronization. Some AJAX frameworks or browser applications use special HTTP requests, which are permanently open in order to retrieve asynchronous data from the server. These requests may let the synchronization hang until the specified synchronization timeout expires. To prevent this situation, either use the HTML synchronization mode or specify the URL of the problematic request in the **Synchronization exclude list** setting.
			Type the entire URL or a fragment of the URL, such as *http://test.com/timeService* or *timeService*.
			Separate entries by comma, for example:
			`'VB .NET code`
			`Agent.SetOption(OPT_XBROWSER_SYNC_EXCLUDE_URLS,`
			`{ "fpdownload.macromedia.com",`
			`"fpdownload.adobe.com",`
			`"download.microsoft.com"`
			`})`

Open Agent Port Numbers

When the Open Agent starts, a random available port is assigned to Silk Test Workbench and to the application that you are testing. The port numbers are registered on the information service. Silk Test Workbench contacts the information service to determine the port to use to connect to the Open Agent. The information service communicates the appropriate port, and Silk Test Workbench connects to that port. Communication runs directly between Silk Test Workbench and the agent.

By default, the Open Agent communicates with the information service on port 22901. You can configure additional ports for the information service as alternate ports that work when the default port is not available. By default, the information service uses ports 2966, 11998, and 11999 as alternate ports.

Typically, you do not have to configure port numbers manually. However, if you have a port number conflict or an issue with a firewall, you must configure the port number for that machine or for the information service. You can use a different port number for a single machine or you can use the same available port number for all your machines.

Configuring the Port to Connect to the Information Service

Before you begin this task, stop the Silk Test Open Agent.

Typically, you do not have to configure port numbers manually. The information service handles port configuration automatically. Use the default port of the information service to connect to the Agent. Then, the information service forwards communication to the port that the Agent uses.

The default port of the information service is 22901. When you can use the default port, you can type `hostname` without the port number for ease of use. If you do specify a port number, ensure that it matches the value for the default port of the information service or one of the additional information service ports. Otherwise, communication will fail.

If necessary, you can change the port number that all clients use to connect to the information service.

1. Navigate to the `infoservice.properties.sample` file and open it.
   - In a Microsoft Windows system, this file is located in `C:\ProgramData\Silk\Silk Test\conf`, where "`C:\ProgramData`" is equivalent to the content of the ALLUSERSPROFILE environment variable, which is set by default on Windows systems.
   - On macOS, this file is located in `/Users/<user>/.silk/silktest/conf`.

   This file contains commented text and sample alternate port settings.
2. Change the value for the appropriate port.

   Typically, you configure the information service port settings to avoid problems with a firewall by forcing communication on a specific port.

   Port numbers can be any number from 1 to 65535.

   - `infoservice.default.port` – The default port where the information service runs. By default, this port is set to 22901.
3. Save the file as `infoservice.properties`.
4. Restart the Open Agent, the Silk Test client, and the application that you want to test.

Configuring the Port to Connect to the Open Agent

Before you begin this task, stop the Silk Test Open Agent.

Typically, you do not have to configure port numbers manually. The information service handles port configuration automatically. Use the default port of the information service to connect to the Agent. Then, the information service forwards communication to the port that the Agent uses.

If necessary, change the port number that the Silk Test client or the application that you want to test uses to connect to the Open Agent.

1. Navigate to the `agent.properties.sample` file and open it.

   By default, this file is located at: `%APPDATA%\Silk\SilkTest\conf`, which is typically `C:\Users\<user name>\AppData\Silk\SilkTest\conf` where *<user name>* equals the current user name.
2. Change the value for the appropriate port.

   Typically, you configure port settings to resolve a port conflict.

   Note: Each port number must be unique. Ensure that the port numbers for the Agent differ from the information service port settings.

   Port numbers can be any number from 1 to 65535.

   Port settings include:

   - `agent.vtadapter.port` – Controls communication between Silk Test Workbench and the Open Agent when running tests.
   - `agent.xpmodule.port` – Controls communication between Silk Test Classic and the Agent when running tests.
   - `agent.autcommunication.port` – Controls communication between the Open Agent and the application that you are testing.
   - `agent.rmi.port` – Controls communication with the Open Agent and Silk4J.
   - `agent.ntfadapter.port` – Controls communication with the Open Agent and Silk4NET.

   Note: The ports for Apache Flex testing are not controlled by this configuration file. The assigned port for Flex application testing is 6000 and increases by 1 for each Flex application that is tested. You cannot configure the starting port for Flex testing.
3. Save the file as `agent.properties`.
4. Restart the Open Agent, the Silk Test client, and the application that you want to test.

Object Maps

An object map is a test asset that contains items that associate a logical name (an alias) with a control or a window, rather than the control or window's locator. Once a control is registered in an object map asset, all references to it in visual tests and scripts are made by its alias, rather than by its actual locator name.

You can use object maps to store objects that you are using often in multiple scripts. Multiple tests can reference a single object map item definition, which enables you to update that object map definition once and have Silk Test Workbench update it in all tests that reference the object map definition.

In your scripts, you can mix object map identifiers and locators. This feature enables you to keep your object maps relatively small and easier to manage. You can simply store the commonly used objects in your object maps, and use locators to reference objects that are rarely used.

Tip: To optimally use the functionality that object maps provide, create an individual project in Silk Test Workbench for each application that you want to test.

Example for object maps

The following construct shows a definition for a `BrowserWindow` where the locator is used:

```
_desktop.BrowserApplication("cnn_com").BrowserWindow("//
BrowserWindow[1]")
```

The name of the object map asset is *cnn_com*. The locator that can be substituted by an alias in the object map is the following:

```
"//BrowserWindow[1]"
```

The object map entry for the `BrowserWindow` is *BrowserWindow*.

The resulting definition of the `BrowserWindow` in the script is the following:

```
_desktop.BrowserApplication("cnn_com").BrowserWindow("BrowserWin
dow")
```

If the index in the locator changes, you can just change the alias in the object map, instead of having to change every appearance of the locator in your test script. Silk Test Workbench will update all tests that reference the object map definition.

Example for mixing object map identifiers and locators

The following sample code shows how you can mix object map identifiers and locators to specify a rarely used child object of an object stored in an object map:

```
// VB
Window window = _desktop.Window("MyApplication") // object map
id - the application window is used often
MenuItem aboutMenuItem =
_desktop.MenuItem("@caption='About'") // locator - the About
dialog is only used once
aboutMenuItem.Select()
```

Advantages of Using Object Maps

Object maps have the following advantages:

- They simplify test maintenance by applying changes made to a locator for an object map item to all tests that include the corresponding object map item.
- They ease the handling of locators in a large scale functional testing environment.
- They can be managed independent of individual scripts.
- They substitute complex locator names with descriptive names, which can make visual tests and scripts easier to read.
- They eliminate dependence on locators, which may change if the test application is modified.

Turning Object Maps Off and On

You can configure Silk Test Workbench to use the locator name or the alias from the object map during recording.

To use the alias from the object map during recording:

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **Locator**.
4. To define whether you want to record object map entries or XPath locators, select the appropriate recording mode from the **Record object map entries** list:
   - **Object map entries for new and existing objects**. This is the default mode.
   - **XPath locators for new and existing objects**.
   - **XPath locators for new objects only**. For objects that already exist in an object map, the object map entry is reused. Choosing this setting enables you to create object maps for the main controls of an AUT, and to persist these object maps while creating additional tests against the AUT.

Note: In addition to the XPath attributes, Silk Test Workbench uses additional attributes of the element when merging object maps during locator recording. However, attributes that might lead to ambiguous usage of object map IDs in a recorded script are not used to map locators to existing object map entries.

Note: When you enable the **Record object maps** setting, object map item names display in place of locators throughout Silk Test Workbench. For instance, if you view the **Application Configurations** category in the **Properties** pane, you will notice that the **Locator** box shows the object map item name rather than the locator name.

Object Map Scope

When you record additional visual tests or scripts, Silk Test Workbench searches the current project to find matching object map items to use when recording. If no matching items are found, Silk Test Workbench searches all referenced projects. If matching items are found in a referenced project, those items are used. If no matching items are found in a referenced project, Silk Test Workbench creates the new items in the current project.

Note: The only exception to this rule occurs if matching items exist in a referenced project. If a new object is identified, it is added to the object map in the referenced project if it is a child of an object map in the referenced project.

To use only the current project with object maps, you can set the **Object map scope** option to **Current project** in the **Options** dialog box. With this setting, only the current project is used to find matching object map items and for recording new items.

Merging Object Maps During Action Recording

When you record actions with Silk Test Workbench, Silk Test Workbench checks if existing object map entries can be reused. Silk Test Workbench checks this directly during recording, when a new locator is generated. Silk Test Workbench checks if the object that is currently recorded in the application under test exactly matches an existing object map entry, and if yes, Silk Test Workbench reuses the object map identifier from the object map.

This behavior has the following benefits:

- Silk Test Workbench correctly reuses an object map identifier during recording, even if the locator in the object map has changed.
- A recorded script cannot contain wrong object map identifiers, and therefore will never fail to play back because of a wrong object map identifier.
- If you restructure your object map, for example by adding an additional level of hierarchy, the object map identifiers are still reused.

Example

Silk Test Workbench records the following script when you click on the **Products** link in the Micro Focus website, *http://www.borland.com*.

```
With _desktop.BrowserApplication( "borland_com" )
  With .BrowserWindow( "BrowserWindow" )
    .DomLink( "Products" ).Click( MouseButton .Left, New Point
(47, 18))
  End With
End With
```

The recorded object map looks like this:

```
borland_com                                  //BrowserApplication
  BrowserWindow                                      //BrowserWindow
    Products                                           //
A[@textContents='Products']
```

You could now manually restructure the object map to include the header section of the Micro Focus website:

```
borland_com                                  //BrowserApplication
  BrowserWindow                                      //BrowserWindow
    header                                             //
HEADER[@role='banner']
      Products                                           //
A[@textContents='Products']
```

When you now record a click on the **Products** link the object map is reused correctly, and the following script is recorded:

```
With _desktop.BrowserApplication( "borland_com" )
  With .BrowserWindow( "BrowserWindow" )
    .DomElement("header").DomLink( "Products" ).Click( MouseButt
on .Left, New Point (47, 18))
  End With
End With
```

Note: When you record another object in the header section of the Micro Focus website, for example the **About** link, Silk Test Workbench adds the **About** object map entry as a child of **BrowserWindow**, and not of **header**.

Using Object Maps with Web Applications

By default, when you record actions against a Web application, Silk Test Workbench creates an object map with the name *WebBrowser* for native browser controls and an object map asset for every Web domain in the *Common* project.

You can change the **Object map location** to the current project in the **General** section of the **Recording** options. Silk Test Workbench will then generate an object map asset for every Web domain during recording.

For common browser controls which are not specific for a Web domain, like the main window or the dialog boxes for printing or settings, an additional object map is generated in the current project with the name *WebBrowser*. You can review the generated object map assets in the **Asset Browser**.

In the object map, you can edit the URL pattern by which the object map entries are grouped. When you edit the pattern, Silk Test Workbench performs a syntactical validation of the pattern. You can use the wildcards * and ? in the pattern.

Example

When you record some actions on *http://www.borland.com* and *http://www.microfocus.com* and then open the printer dialog, the following three new object map assets are added to the **Asset Browser**:

- WebBrowser
- borland_com
- microfocus_com

Note: Silk Test Workbench generates the new object map assets only for projects without an object map. If you record actions against a Web application for which Silk Test Workbench already includes an object map that was generated with a version of Silk Test Workbench prior to version 14.0, the additionally recorded entries are stored into the existing object map, and there are no additional object map assets generated for the Web domains.

Renaming an Object Map Item

You can manually rename items and locators in an object map.

Warning: Renaming an object map item affects every visual test or script that uses that item. For example, if you rename the **Cancel** button object map item from **CancelMe** to **Cancel**, every script that uses **CancelMe** must be changed manually to use **Cancel**.

Object map items must be unique. If you try to add a duplicate object map item, Silk Test Workbench notifies you that the object must be unique.

If you use an invalid character or locator, the item name or locator text displays in red and a tooltip explains the error. Invalid characters for object map items include: \, /, <, >, ", :, *, ?, |, =, ., @, [,]. Invalid locator paths include: empty or incomplete locator paths.

When you rename the root node of the object map, the object map asset is also renamed. You cannot save an object map asset if you have changed the name of the root node to the name of an existing object map asset. When you change the object map asset name, the root node of the object map is also renamed.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. From the **Asset Types** list, select **Object Map**. Any existing object maps for the selected project display in the right pane.

Tip: If you do not see the object map that you want to edit, ensure that the correct project is selected in the **Active Project** list.

3. Choose one of the following:

   - Double-click the object map that includes the object map item that you want to rename.
   - Right-click the object map that includes the object map item that you want to rename and choose **Open**.

   The object map displays a hierarchy of the object map items and the locator associated with each item.

   Note: If another user is currently editing the object map that you select, a message box notifies you and names the user who currently has the object map locked.

4. Click **Edit**.

   By default, the object map is read only. Clicking **Edit** gives you read/write access and ensures that no other users can edit the entries while you are modifying them. You can also press **F2** to edit the currently selected field in the object map.

   The **Edit** button changes to **Revert**. If you want to undo any changes that you make during this session, click **Revert**.

5. Navigate to the object map item that you want to rename.
   For example, you might need to expand a node to locate the item that you want to rename.

6. Click the object that you want to rename and then click the object again or press **F2**.

   Once you access in-line editing mode, you can click an object one time to edit it. Press **Tab** to toggle between the name and the description of the object map item.

   A black box displays around the item field and the text is highlighted.

7. Type the item name that you want to use and then press `Enter`.

   If you use an invalid character, the item name displays in red.

   The new name displays in the **Item name** list.

8. Perform one of the following steps:

   - Click **Save** to save your changes and continue working on the object map.

     Silk Test Workbench saves your changes as a new version and the object map mode goes back to read only.
   - Click **Save and close** to save your changes and exit the object map.

     Silk Test Workbench saves your changes as a new version and the object map is available for other users to modify.

Note: All child nodes of any node in the object map tree are sorted alphabetically when you save the object map.

If any existing visual tests or scripts use the item name that you changed, you must manually change the visual tests or scripts to use the new item name.

Note: While recording against a web application or a mobile web app, you can directly change the name of the object map entry in the **Choose Action** dialog. Right-click on the object and then expand the **Object identification** area of the **Choose Action** dialog. Then you can edit the object map entry in the **Object Map ID** field. This functionality is available if you are testing against one of the following browsers:

- Microsoft Edge.
- Apple Safari.
- Mozilla Firefox 41 or later.
- Google Chrome 50 or later.
- A mobile browser.

Modifying Object Maps

By default, object maps are read only. When you record a visual test or script that uses object maps, object map items are added to the database as assets.

When you finish recording a visual test or script that uses an object map, Silk Test Workbench determines if another user is using that object map. If another user is modifying the object map, a message box notifies you that the object map is being modified by another user and provides the name of that user. To prevent locked object maps from prohibiting recording, you can set the **Record locators for locked object maps** option to **Yes** in the **Options** dialog box. With this setting, when an object map is being modified by another user, any previously identified object map items are used where applicable and locators are used for any new objects that you record.

An existing object map is able to reuse existing object map identifiers during recording, even if you have added additional structural elements to the object map.

Example: Adding a DIV to an existing object map

Let us suppose you want to add a DIV element to bundle the email and login fields in the following simple object map:

```
demo_borland_com                                    //
BrowserApplication
  BrowserWindow                                              //
BrowserWindow
    login-form email                                         //
INPUT[@id='login-form:email']
    login-form login                                         //
INPUT[@id='login-form:login']
```

You can change the structure of the object map by adding the new DIV *loginArea* and the object map will still be able to correctly reuse the object map identifiers during recording.

```
demo_borland_com                                    //
BrowserApplication
  BrowserWindow                                              //
BrowserWindow

loginArea                                                 //'DIV[@id='
login']
      login-form email                                       //
INPUT[@id='login-form:email']
      login-form login                                       //
INPUT[@id='login-form:login']
```

Modifying a Locator in an Object Map

Locators are automatically associated with an object map item when you record a visual test or script. However, you might want to modify a locator path to make it more generic. For example, if your test application automatically assigns the date or time to a specific control, you might want to modify the locator for that control to use a wildcard. Using a wildcard enables you to use the same locator for each test even though each test inserts a different date or time.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. From the **Asset Types** list, select **Object Map**. Any existing object maps for the selected project display in the right pane.

Tip: If you do not see the object map that you want to edit, ensure that the correct project is selected in the **Active Project** list.

3. Choose one of the following:

   - Double-click the object map that includes the locator that you want to modify.
   - Right-click the object map that includes the locator that you want to modify and choose **Open**.

   The object map displays a hierarchy of the object map items and the locator associated with each item.

   Note: If another user is currently editing the object map that you select, a message box notifies you and names the user who currently has the object map locked.

4. Click **Edit**.

   By default, the object map is read only. Clicking **Edit** gives you read/write access and ensures that no other users can edit the entries while you are modifying them. You can also press **F2** to edit the currently selected field in the object map.

   The **Edit** button changes to **Revert**. If you want to undo any changes that you make during this session, click **Revert**.

5. Navigate to the locator that you want to modify.
   For example, you might need to expand a node to locate the locator that you want to modify.

6. Click the locator path that you want to modify and then click the locator path again or click **F2**.

   Once you access in-line editing mode, you can click a locator path one time to edit it. Press **Tab** to toggle between the name and the description of the object map item.

   A black box displays around the locator path field and the text is highlighted.

7. If you have a valid locator path, you can type the item name and locator path that you want to use and then press `Enter`. To determine a valid locator path, use the **Identify Object** dialog box as described in the following steps:

   a) Choose **Tools** > **Identify Object**.

      The **Select Application** dialog box might open if the test application has not been associated with the object map. If this happens, select the application that you want to test and then click **OK**.

      The **Identify Object** dialog box opens.

   b) Specify the **Selection mode**.

      - **Click** – Click the object to identify the locator.
      - **Hot Key** – Specify this mode to capture the object using the keystroke combination specified in the **Keystroke** list box. Typically, you choose this mode to capture objects, such as a menu or combobox, that only appear when clicked by the user. With this mode, you select the object and then press the hot key keystroke combination to capture the locator without dismissing the object.

   c) Click **Start Identify**.

   d) Position the mouse over the object that you want to record and perform one of the following steps:

      - If you use **Click** mode, click the object that you want to identify.
      - Press the keystroke combination to capture the object with the **Hot Key** mode.

        By default, the keystroke combination is `Ctrl+Shift`.

      Silk Test Workbench lists the related locator string in the **Selected Locator** text box.

   e) Select the locator that you want to use in the **Locator Details** table. The new locator displays in the **Selected Locator** text box.

   f) Click **Paste**. Silk Test Workbench adds an item name and the associated locator to the object map.

8. If necessary, modify the item name or locator text to meet your needs.

   If you use an invalid character or locator, the item name or locator text displays in red and a tooltip explains the error.

   Invalid characters for object map items include: \, /, <, >, ", :, *, ?, |, =, ., @, [,].

   Invalid locator paths include: empty or incomplete locator paths.

9. Perform one of the following steps:
   - Click **Save** to save your changes and continue working on the object map.

     Silk Test Workbench saves your changes as a new version and the object map mode goes back to read only.
   - Click **Save and close** to save your changes and exit the object map.

     Silk Test Workbench saves your changes as a new version and the object map is available for other users to modify.

If any existing visual tests or scripts use the locator path that you modified, you must manually change the visual tests or scripts to use the new locator path.

Updating Object Maps from the Test Application

If items in the test application change, you can use the **Object Map** UI to update the locators for these items.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. From the **Asset Types** list, select **Object Map**. Any existing object maps for the selected project display in the right pane.

   Tip: If you do not see the object map that you want to edit, ensure that the correct project is selected in the **Active Project** list.
3. Choose one of the following:
   - Double-click the object map that you want to use.
   - Right-click the object map that you want to use and choose **Open**.

   The object map displays a hierarchy of the object map items and the locator associated with each item.

   Note: If another user is currently editing the object map that you select, a message box notifies you and names the user who currently has the object map locked.
4. Click **Edit**.

   By default, the object map is read only. Clicking **Edit** gives you read/write access and ensures that no other users can edit the entries while you are modifying them. You can also press **F2** to edit the currently selected field in the object map.

   The **Edit** button changes to **Revert**. If you want to undo any changes that you make during this session, click **Revert**.
5. Click **Update Locator**. The **Identify Object** dialog box displays and Silk Test Workbench opens the test application.
6. Position the mouse cursor over the object that you want to record and press **CTRL+ALT**. Silk Test Workbench displays the locator string in the **Locator** text field.
7. Select the locator that you want to use in the **Locator Details** table.
8. Remove any attributes that you do not want to use from the locator that is displayed in the **Locator** text field.
9. Click **Validate Locator** to validate that the locator works.
10. Click **Paste Locator to Editor** to update the locator in the object map.
11. Save the changed object map.

When you update an object map item from the AUT, you can change only the XPath representations of leaf nodes in the object map tree. You cannot change the XPath representations of any parent nodes. When the XPath representations of higher-level nodesin the object map tree are not consistent after the update, an error message displays.

Example

For example, suppose you have an object map item with an object map ID that has the following three hierarchy levels:

```
WebBrowser.Dialog.Cancel
```

The corresponding XPath representation of these hierarchy levels is the following:

```
/BrowserApplication//Dialog//PushButton[@caption='Cancel']
```

- First hierarchy level: `/BrowserApplication`
- Second hierarchy level: `//Dialog`
- Third hierarchy level: `//PushButton[@caption='Cancel']`

You can use the following locator to update the object map item:

```
/BrowserApplication//Dialog//PushButton[@id='123']
```

- First hierarchy level: `/BrowserApplication`
- Second hierarchy level: `//Dialog`
- Third hierarchy level: `//PushButton[@id='123']`

You cannot use the following locator cannot to update the object map item, because the second level hierarchy nodes do not match:

```
/BrowserApplication//BrowserWindow//PushButton[@id='9999999']
```

- First hierarchy level: `/BrowserApplication`
- Second hierarchy level: `//BrowserWindow`
- Third hierarchy level: `//PushButton[@id='9999999']`

Copying an Object Map Item

You can copy and paste object map entries within or between object maps. For example, if the same functionality exists in two separate test applications, you might copy a portion of one object map into another object map.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. From the **Asset Types** list, select **Object Map**. Any existing object maps for the selected project display in the right pane.

   Tip: If you do not see the object map that you want to edit, ensure that the correct project is selected in the **Active Project** list.
3. Choose one of the following:

   - Double-click the object map that includes the object map item that you want to copy.
   - Right-click the object map that includes the object map item that you want to copy and choose **Open**.

   The object map displays a hierarchy of the object map items and the locator associated with each item.

   Note: If another user is currently editing the object map that you select, a message box notifies you and names the user who currently has the object map locked.
4. Click **Edit**.

   By default, the object map is read only. Clicking **Edit** gives you read/write access and ensures that no other users can edit the entries while you are modifying them. You can also press **F2** to edit the currently selected field in the object map.

The **Edit** button changes to **Revert**. If you want to undo any changes that you make during this session, click **Revert**.

5. Navigate to the object map item that you want to copy.
   For example, you might need to expand a node to locate the item that you want to copy.
6. Choose one of the following:
   - Right-click the object map item that you want to copy and choose **Copy tree**.
   - Click the object map item that you want to copy and then choose **Edit** > **Copy**.
   - Click the object map item that you want to copy and then press `Ctrl+C`.
7. In the object map hierarchy, navigate to the position where you want to paste the item that you copied.
   For instance, to include an item on the first level of the hierarchy, click the first item name in the item list. To position the copied item a level below a specific item, click the item that you want to position the copied item below.
   To copy and paste between object maps, you must exit the map where you copied the object map item and open and edit the object map where you want to paste the object map item.
8. Choose one of the following:
   - Right-click the position in the object map where you want to paste the copied object map item and choose **Paste**.
   - Click the position in the object map where you want to paste the copied object map item and then choose **Edit** > **Paste**.
   - Click the position in the object map where you want to paste the copied object map item and then press `Ctrl+V`.

   To copy and paste between object maps, ensure that you click **Edit** before you attempt to paste the object map item.
   The object map item displays in its new position in the hierarchy.
9. Perform one of the following steps:
   - Click **Save** to save your changes and continue working on the object map.
     Silk Test Workbench saves your changes as a new version and the object map mode goes back to read only.
   - Click **Save and close** to save your changes and exit the object map.
     Silk Test Workbench saves your changes as a new version and the object map is available for other users to modify.

If any existing visual tests or scripts use the object map item name that you moved, you must manually change the visual tests or scripts to use the new position in the hierarchy.

Adding an Object Map Item

Object map items are automatically created when you record a visual test or script. Occasionally, you might want to manually add an object map item.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. From the **Asset Types** list, select **Object Map**. Any existing object maps for the selected project display in the right pane.

   Tip: If you do not see the object map that you want to edit, ensure that the correct project is selected in the **Active Project** list.
3. Double-click the object map to which you want to add the new item. The object map displays a hierarchy of the object map items and the locator associated with each item.

Note: If another user is currently editing the object map that you select, a message box notifies you and names the user who currently has the object map locked.

4. Click **Edit**.

   By default, the object map is read only. Clicking **Edit** gives you read/write access and ensures that no other users can edit the entries while you are modifying them. You can also press **F2** to edit the currently selected field in the object map.

   The **Edit** button changes to **Revert**. If you want to undo any changes that you make during this session, click **Revert**.

5. In the object map hierarchy, right-click on the item below which you want to add the new object map item.

   For instance, to include an item on the first level of the hierarchy, right-click on the first item name in the item list. To position the new item a level below a specific item, right-click on the item below which you want to position the new item.

6. Click **Insert new**. A new item is added to the hierarchy, as the first child of the current node.

7. If you have a valid locator path, you can type the item name and locator path that you want to use and then press `Enter`. To determine a valid locator path, use the **Identify Object** dialog box as described in the following steps:

   a) Choose **Tools** > **Identify Object**.

      The **Select Application** dialog box might open if the test application has not been associated with the object map. If this happens, select the application that you want to test and then click **OK**.

      The **Identify Object** dialog box opens.

   b) Specify the **Selection mode**.

      - **Click** – Click the object to identify the locator.
      - **Hot Key** – Specify this mode to capture the object using the keystroke combination specified in the **Keystroke** list box. Typically, you choose this mode to capture objects, such as a menu or combobox, that only appear when clicked by the user. With this mode, you select the object and then press the hot key keystroke combination to capture the locator without dismissing the object.

   c) Click **Start Identify**.

   d) Position the mouse over the object that you want to record and perform one of the following steps:

      - If you use **Click** mode, click the object that you want to identify.
      - Press the keystroke combination to capture the object with the **Hot Key** mode.

        By default, the keystroke combination is `Ctrl+Shift`.

      Silk Test Workbench lists the related locator string in the **Selected Locator** text box.

   e) Select the locator that you want to use in the **Locator Details** table. The new locator displays in the **Selected Locator** text box.

   f) Click **Paste**. Silk Test Workbench adds an item name and the associated locator to the object map.

8. If necessary, modify the item name or locator text to meet your needs.

   If you use an invalid character or locator, the item name or locator text displays in red and a tooltip explains the error.

   Invalid characters for object map items include: \, /, <, >, ", :, *, ?, |, =, ., @, [,].

   Invalid locator paths include: empty or incomplete locator paths.

9. Perform one of the following steps:

   - Click **Save** to save your changes and continue working on the object map.

     Silk Test Workbench saves your changes as a new version and the object map mode goes back to read only.
   - Click **Save and close** to save your changes and exit the object map.

     Silk Test Workbench saves your changes as a new version and the object map is available for other users to modify.

Note: All child nodes of any node in the object map tree are sorted alphabetically when you save the object map.

Opening an Object Map from a Script

When you are editing a script, you can open an object map by right clicking on an object map entry in the script and selecting **Open Asset.** This will open the object map in the GUI.

Example

```
// VB .NET code
<TestMethod()> Public Sub TestMethod1()
  With _desktop.Window("Untitled -
Notepad").TextField("TextField").TypeKeys("hello")
  End With
End Sub
```

In the previous code sample, right-click `Untitled - Notepad` to open the entry `Untitled - Notepad` in the object map, or right-click `TextField` to open the entry `Untitled - Notepad.TextField` in the object map.

Locating an Object Map Item in the Test Application

After you add or record an object map item, you can click **Locate** to highlight the item in the test application. You might want to highlight an item to confirm that it's the item that you want to modify in the object map.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. From the **Asset Types** list, select **Object Map**. Any existing object maps for the selected project display in the right pane.

   Tip: If you do not see the object map that you want to edit, ensure that the correct project is selected in the **Active Project** list.
3. Choose one of the following:

   - Double-click the object map that you want to use.
   - Right-click the object map that you want to use and choose **Open**.

   The object map displays a hierarchy of the object map items and the locator associated with each item.

   Note: If another user is currently editing the object map that you select, a message box notifies you and names the user who currently has the object map locked.
4. Click **Edit**.

   By default, the object map is read only. Clicking **Edit** gives you read/write access and ensures that no other users can edit the entries while you are modifying them. You can also press **F2** to edit the currently selected field in the object map.

   The **Edit** button changes to **Revert**. If you want to undo any changes that you make during this session, click **Revert**.
5. In the object map hierarchy, select the object map item that you want to highlight in the test application.

   Note: Ensure that only one instance of the test application is running. Running multiple instances of the test application will cause an error because multiple objects will match the locator.
6. Click **Locate**.

   The **Select Application** dialog box might open if the test application has not been associated with the object map. If this happens, select the application that you want to test and then click **OK**.

If you select the incorrect application in the **Select Application** dialog box, click **Change Application and Locate** from the **Locate** submenu. The **Select Application** dialog box opens and you can select the correct test application.

Silk Test Workbench opens the test application and displays a green box around the control that the object map item represents.

Finding Errors in an Object Map

If you use an invalid character or locator, the item name or locator text displays in red and a tooltip explains the error. Use the toolbar in the **Object Map** window to navigate to any errors.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. From the **Asset Types** list, select **Object Map**. Any existing object maps for the selected project display in the right pane.

   Tip: If you do not see the object map that you want to edit, ensure that the correct project is selected in the **Active Project** list.
3. Choose one of the following:

   - Double-click the object map that you want to troubleshoot.
   - Right-click the object map that you want to troubleshoot and choose **Open**.

   The object map displays a hierarchy of the object map items and the locator associated with each item.

   Note: If another user is currently editing the object map that you select, a message box notifies you and names the user who currently has the object map locked.
4. Click **Edit**.

   By default, the object map is read only. Clicking **Edit** gives you read/write access and ensures that no other users can edit the entries while you are modifying them. You can also press **F2** to edit the currently selected field in the object map.

   The **Edit** button changes to **Revert**. If you want to undo any changes that you make during this session, click **Revert**.
5. Look for any item name or locator text displayed in red.
6. If necessary, modify the item name or locator text to meet your needs.

   If you use an invalid character or locator, the item name or locator text displays in red and a tooltip explains the error.

   Invalid characters for object map items include: \, /, <, >, ", :, *, ?, |, =, ., @, [,].

   Invalid locator paths include: empty or incomplete locator paths.
7. Perform one of the following steps:

   - Click **Save** to save your changes and continue working on the object map.

     Silk Test Workbench saves your changes as a new version and the object map mode goes back to read only.
   - Click **Save and close** to save your changes and exit the object map.

     Silk Test Workbench saves your changes as a new version and the object map is available for other users to modify.

Deleting an Object Map Item

You might want to delete an item from an object map if it no longer exists in the test application or for some other reason.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. From the **Asset Types** list, select **Object Map**. Any existing object maps for the selected project display in the right pane.

   Tip: If you do not see the object map that you want to edit, ensure that the correct project is selected in the **Active Project** list.
3. Double-click the object map that includes the object map item that you want to delete. The object map displays a hierarchy of the object map items and the locator associated with each item.

   Note: If another user is currently editing the object map that you select, a message box notifies you and names the user who currently has the object map locked.
4. Click **Edit**.

   By default, the object map is read only. Clicking **Edit** gives you read/write access and ensures that no other users can edit the entries while you are modifying them. You can also press **F2** to edit the currently selected field in the object map.

   The **Edit** button changes to **Revert**. If you want to undo any changes that you make during this session, click **Revert**.
5. Navigate to the object map item that you want to delete.
   For example, you might need to expand a node to locate the object map item that you want to delete.
6. Choose one of the following:

   - Right-click the object map item that you want to delete and choose **Delete**, or choose **Delete tree** to additionally delete all child items of the object map item.
   - Click the object map item that you want to delete and then press **DEL**, or press **CTRL+DEL** to additionally delete all child items of the object map item.

   After deleting an object map item, the focus moves to the next item in the object map.
7. Perform one of the following steps:

   - Click **Save** to save your changes and continue working on the object map.

     Silk Test Workbench saves your changes as a new version and the object map mode goes back to read only.
   - Click **Save and close** to save your changes and exit the object map.

     Silk Test Workbench saves your changes as a new version and the object map is available for other users to modify.

If any existing visual tests or scripts use the object map item or its children that you deleted, you must manually change any references to that object map item in the visual tests or scripts.

Initially Filling Object Maps

As a best practice, we recommend that you fill your object map and then review all object map items before you record your tests.

To initially fill your object map with all available items in the AUT, you might create a visual test that clicks every object and opens every window and dialog box in your test application. Then, you can review the object map item for each object and make any necessary modifications before you record your functional tests. After you have reviewed and modified the object map items you can delete the visual test that you have created to fill the object map.

Tip: You can use the arrow keys to navigate between items in an object map.

Grouping Elements in Object Maps

When items in an object map have no consistent parent object, you can group these elements by adding a new tree item with the locator ".", which is the locator for the current element in Xpath.

Warning: Grouping object map items affects every visual test or script that uses these items. Every visual test or script that uses these items must be changed manually to use the new locators.

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. From the **Asset Types** list, select **Object Map**. Any existing object maps for the selected project display in the right pane.

   Tip: If you do not see the object map that you want to edit, ensure that the correct project is selected in the **Active Project** list.
3. Choose one of the following:

   - Double-click the object map that you want to edit.
   - Right-click the object map that you want to edit and choose **Open**.

   The object map displays a hierarchy of the object map items and the locator associated with each item.

   Note: If another user is currently editing the object map that you select, a message box notifies you and names the user who currently has the object map locked.
4. Click **Edit**.

   By default, the object map is read only. Clicking **Edit** gives you read/write access and ensures that no other users can edit the entries while you are modifying them. You can also press **F2** to edit the currently selected field in the object map.

   The **Edit** button changes to **Revert**. If you want to undo any changes that you make during this session, click **Revert**.
5. Right click on the tree item below which you want to add the new structuring item and choose **Insert New**.
6. Double click the **Item name** field of the new object map item.
7. Type the item name that you want to use and then press `Enter`.

   If you use an invalid character, the item name displays in red.

   The new name displays in the **Item name** list.
8. Click the **Locator path** field of the new object map item and type `.` into the field.
9. Press `Enter`.
10. For every object map item that you want to relocate to a new location under the new item:
    a) Right click on the item that you want to relocate and choose **Cut tree**.
    b) Right click on the new structuring item and choose **Paste**.
11. Perform one of the following steps:

    - Click **Save** to save your changes and continue working on the object map.

      Silk Test Workbench saves your changes as a new version and the object map mode goes back to read only.
    - Click **Save and close** to save your changes and exit the object map.

      Silk Test Workbench saves your changes as a new version and the object map is available for other users to modify.

Object Maps: Frequently Asked Questions

This section lists questions that you might encounter when using object maps with Silk Test Workbench.

Can I Merge Multiple Object Maps Into a Single Map?

Although Micro Focus recommends recording into the same object map instead of merging existing object maps, you can use a text editor to merge multiple object maps into a single map.

What Happens to an Object Map when I Delete a Test Script?

When you delete a test script or a visual test that includes object map entries, the associated object maps are not changed. All object map entries are persisted.

Can I Manually Create an Object Map for My Application Under Test?

Micro Focus recommends creating object maps during the recording of a test. However, you can also create an empty object map and manually add object map entries to this map.

To create a new object map, right-click on **Object map** in the **Asset Browser** and select **New Object map**.

Image Recognition Support

You can use image recognition in the following situations:

- To conveniently interact with test applications that contain highly customized controls, which cannot be identified using object recognition. You can use *image clicks* instead of coordinate-based clicks to click on a specified image.
- To test graphical objects in the application under test, for example charts.
- To perform a check of the visible UI of the application under test.

If you want to click on a control that is otherwise not recognizable, you can use the `ImageClick` method with an image asset. If you want to verify that an otherwise not recognizable control exists in your application under test, you can use the `VerifyAsset` method with an image verification.

Image recognition methods are supported for all technology domains that are supported by Silk Test Workbench.

Note: Image recognition methods do not work with controls that are not visible on the screen. For example, you cannot use image recognition for an image that is scrolled out of view.

Image Click Recording

Image click recording is disabled by default in favor of coordinate-based click recording, because image click recording might generate a confusingly large number of images.

To enable image click recording, click **Tools** > **Options** > **Record** > **General** and set the value of **Record image clicks** to `Yes`.

Note: When recording on a mobile browser, you do not have to enable image click recording.

When image click recording is enabled, Silk Test Workbench records `ImageClick` methods when object recognition or text recognition is not possible. You can insert image clicks in your script for any control, even if the image clicks are not recorded.

If you do not whish to record an `ImageClick` action, you can turn off image click recording and record normal clicks or text clicks.

Note: The recorded images are not reused. Silk Test Workbench creates a new image asset for each image click that you record.

Note: Image click recording is not supported for applications or applets that use the Java AWT/Swing controls.

Image Recognition Methods

Silk Test Workbench provides the following methods for image recognition:

Method	Description
`ImageClick`	Clicks in the middle of the image that is specified in an asset. Waits until the image is found or the *Object resolve timeout*, which you can define in the synchronization options, is over.
`ImageExists`	Returns whether the image that is specified in an asset exists.

Method	Description
`ImageRectangle`	Returns the object-relative rectangle of the image that is specified in an asset.
`ImageClickFile`	Clicks on the image that is specified in a file.
`ImageExistsFile`	Returns whether the image that is specified in a file exists.
`ImageRectangleFile`	Returns the object-relative rectangle of the image that is specified in a file.
`VerifyAsset`	Executes a verification asset. Throws a VerificationFailedException if the verification does not pass.

Note: Image recognition methods do not work with controls that are not visible on the screen. For example, you cannot use image recognition for an image that is scrolled out of view.

Image Assets

You can use image assets in the following situations:

- To conveniently interact with test applications that contain highly customized controls, which cannot be identified using object recognition. You can use *image clicks* instead of coordinate-based clicks to click on a specified image.
- To test graphical objects in the application under test, for example charts.

Image assets consist of an image with some additional information that is required by Silk Test Workbench to work with the asset.

Silk Test Workbench provides the following methods for image assets:

Method	Description
`ImageClick`	Clicks in the middle of the specified image asset. Waits until the image is found or the *Object resolve timeout*, which you can define in the synchronization options, is over.
`ImageExists`	Returns whether the specified image asset exists.
`ImageRectangle`	Returns the object-relative rectangle of the specified image asset.

Creating an Image Asset

You can create image assets in one of the following ways:

- By inserting a new image asset into an existing visual test or script.
- During recording.
- By using the **Asset Browser**.

To create a new image asset in the **Asset Browser**, perform the following steps:

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. In the menu, click **New Image Asset**.
3. In the **Asset Types** list, right-click **Image Asset** and select **New Image Asset**. The image asset UI opens.
4. Select how you want to add an image to the asset.
   - If you want to use an existing image, click **Browse** and select the image file.

- If you want to capture a new image from the UI of the application under test, click **Capture**. If you are testing a Web application, you can select the browser on which you want to capture the image from the **Select Application** window.

5. If you have selected to capture a new image, select the area of the screen that you want to capture and click **Capture Selection**.
6. In the **Name** text field, type a meaningful name for the new image asset.

   Note: If you have selected to use an existing image, the name of the image asset is by default the name of the existing image.
7. *Optional:* Click **Verify** to check if Silk Test Workbench can find the image asset in the UI of the AUT.

   If you are testing a Web application, you can select the browser on which you want to capture the image from the **Select Application** window.
8. *Optional:* Check the **Click position** check box to select the location on which any clicks on the image asset are performed.

   The default location is the center of the image. Type the location into the **x** and **y** fields or select the location on the image.
9. Specify the **Accuracy Level**.

   The accuracy level defines how much the image to be verified is allowed to be different to the image in the application under test, before Silk Test Workbench declares the images as different. This is helpful if you are testing multiple systems or browsers with different screen resolutions. We recommend to choose a high level of accuracy in order to prevent false positives. You can change the default accuracy level in the options.

   Note: When you set the **Accuracy Level** to less than five, the actual colors of the images are no longer considered for the comparison. Only the grayscale representations of the images are compared.
10. Click **Save** to save the image asset into the Silk Test Workbench repository, or click **Save and close** to additionally close the image asset UI.

The new image asset is listed in the **Asset Browser**, and you can use it to perform image clicks.

You can add multiple images to the same image asset.

Note: To add an image click while recording against a mobile browser, you can right-click in the **Recording** window and select **ImageClick** from the action list.

Adding Multiple Images to the Same Image Asset

During testing, you will often need to test functionality on multiple environments and with different testing configurations. In a different environment, the actual image might differ in such a degree from the image that you have captured in the image asset, that image clicks might fail, although the image is existing. In such a case, you can add multiple images to the same image asset.

To add an additional image to an image asset:

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. In the **Asset Types** list, select **Image Asset**
3. Double-click on the image asset to which you want to add an additional image. The image asset UI opens.
4. Click on the plus sign in the lower part of the UI to add a new image to the image asset.
5. Click **Save** to save the image asset into the Silk Test Workbench repository, or click **Save and close** to additionally close the image asset UI.

The new image is added to the asset. Each time an image click is called, and until a match is achieved, Silk Test Workbench will compare the images in the asset with the images in the UI of the application under test. By default, Silk Test Workbench compares the images in the order in which they have been added to the asset.

Note: To change the order in which Silk Test Workbench compares the images, click on an image in the lower part of the image asset UI and drag the image to the position that you want. The order lowers from left to right. The image that is compared first is the image in the left-most position.

Opening an Asset from a Script

When you are editing a script, you can open an asset by right clicking it and selecting **Open Asset.** This will open the asset in the GUI.

If the asset is a reference to a file on the system, for example, referenced by `ImageClickFile`, the file will be opened by your system's default editor.

Using Variables as Image Asset Names in Image Clicks

When executing an image click from a visual test, you can use a visual test variable, an active data variable, or an expression for the image asset name. This enables you to execute the same image click on various image assets, depending on the value of the variable or the expression.

For a VB .NET script, this functionality is supported through the image recognition methods, `ImageClick`, `ImageExists`, and so on. In these methods, you specify a string as the image asset name. This string can be the name of a variable or of an expression.

For a visual test, you need to specify the name for the image asset in the properties of the image click test step:

1. Open the visual test that contains the image click.
2. Select the image click test step.
3. Open the **Properties** pane.
4. Expand the **Parameters** node.
5. In the **Image** field, click **...** and select the variable that you want to use as the image asset name.

Image Verifications

You can use an *Image Verification* to check if an image exists in the UI of the application under test (AUT) or not.

Image verifications consist of an image with some additional information that is required by Silk Test Workbench to work with the asset.

To execute an image verification, use the `VerifyAsset` method.

An image verification fails when Silk Test Workbench cannot find the image in the AUT. In this case the script will continue.

If the locator for the image verification is not found in the AUT, Silk Test Workbench throws an `ObjectNotFoundException`.

Creating an Image Verification

You can create image verifications in one of the following ways:

- By using the **Asset Browser**.
- During recording.
- By inserting a new image verification into an existing visual test.

To create a new image verification in the **Asset Browser**, perform the following steps:

1. Choose **View** > **Asset Browser** or select **Get Started** > **Asset Browser** in the **Start Screen**.
2. In the **Asset Types** list, right-click **Image Verification** and select **New Image Verification**. The image verification UI opens.
3. Click **Identify** to identify the image that you want to verify in the application under test.
4. In the **Name** text field, type a meaningful name for the new image verification.

   Note: If you have selected to use an existing image, the name of the image verification is by default the name of the existing image.
5. *Optional:* If you want to recapture the same image from the application under test, because there is a change in comparison to the image that you had initially captured, click **Recapture**.

   If you are testing a Web application, you can select the browser on which you want to capture the image from the **Select Application** window.
6. *Optional:* You can click **Verify** to test if the image verification works. Silk Test Workbench searches for the image in the UI of the AUT, top-down and left to right, and highlights the first matching image.
7. *Optional:* You can add an exclusion area to the image verification, which will not be considered when Silk Test Workbench compares the image verification to the UI of the application under test (AUT).
8. *Optional:* You can set the option **Client Area Only** to define that only the part of the image that is actually part of the AUT is considered when Silk Test Workbench compares the image verification to the UI of the AUT.
9. Specify the **Accuracy Level**.

   The accuracy level defines how much the image to be verified is allowed to be different to the image in the application under test, before Silk Test Workbench declares the images as different. This is helpful if you are testing multiple systems or browsers with different screen resolutions. We recommend to choose a high level of accuracy in order to prevent false positives. You can change the default accuracy level in the options.

   Note: When you set the **Accuracy Level** to less than five, the actual colors of the images are no longer considered for the comparison. Only the grayscale representations of the images are compared.
10. Click **Save** to save the new image verification into the Silk Test Workbench repository, or click **Save and close** to additionally close the image verification UI.

The new image verification is listed in the **Asset Browser**, and you can use it to check if the image exists in the UI of your application under test.

Adding an Image Verification to a Visual Test

You can add image verifications to your visual tests to check if controls which are otherwise not recognizable exists in the UI of the application under test. To add an image verification to a visual test:

1. Open the visual test.
2. Right-click on the step after which you want to add the image verification.
3. Click **Insert** > **Test Logic** > **Verification**. The **Test Logic Designer** opens.
4. Click **Next**. The **Select a Logic Type** page opens.
5. Click **A verification asset**. The **Build the Verification Asset** page opens.
6. Perform one of the following steps:
   - To create a new image verification in the image verification UI, click **Create a new verification**.
   - To insert an existing image verification asset, click **Insert an Existing Verification**.
7. Click **Next**. The **Summary** page opens and shows you a preview of your new verification.
8. Click **Finish**. The new image verification is added.

Adding an Image Verification During Recording

You can add image verifications to your visual tests or VB .NET scripts to check if controls which are otherwise not recognizable exist in the UI of the application under test. To add an image verification during the recording of a visual test or a VB .NET script, perform the following steps:

1. Begin recording.
2. Move the mouse cursor over the object that you want to verify and press **Ctrl+Alt**.

   When you are recording a mobile Web application, you can also click on the object and click **Add Verification**.

   This option temporarily suspends recording and displays the **Test Logic Designer** wizard.
3. Click **Next**. The **Select a Logic Type** page opens.
4. Click **A verification asset**. The **Build the Verification Asset** page opens.
5. Perform one of the following steps:
   - To create a new image verification in the image verification UI, click **Create a new verification**.
   - To insert an existing image verification asset, click **Insert an Existing Verification**.
6. Click **Next**. The **Summary** page opens and shows you a preview of your new verification.
7. Click **Finish**. The new image verification is added.
8. Continue recording.

Examining Why an Image Verification has Failed

When the execution of an image verification in a visual test or a script fails, you can open the **Image Verification Differences View** to examine the differences between the expected image and the actual image found in the application under test.

To open the **Image Verification Differences View**, perform the following steps:

1. Open the result of the execution run in which the image verification has failed.
2. Right-click on the failed image verification step in the visual test or script. The **Image Verification Differences View** opens.
3. Compare the expected image and the actual image to see why the image verification failed.
4. *Optional:* If an image verification has failed because the actual image in the AUT has changed, click **Save as Expected** to set the actual image as the expected image.

Using Variables as Verification Asset Names in Image Verifications

When executing an image verification from a visual test, you can use a visual test variable, an active data variable, or an expression for the verification asset name. This enables you to execute the same image verification on various verification assets, depending on the value of the variable or the expression.

For a VB .NET script, this functionality is supported through the image recognition methods, `ImageClick`, `ImageExists`, and so on. In these methods, you specify a string as the image asset name. This string can be the name of a variable or of an expression.

For a visual test, you need to specify the name for the verification asset in the properties of the image verification test step:

1. Open the visual test that contains the image verification.
2. Select the image verification test step.

3. Open the **Properties** pane.
4. In the **Name** field, click **...** and select the variable that you want to use as the verification asset name.

Image Verification Differences View

When the execution of an image verification in a visual test or a script fails, the **Image Verification Differences View** displays the differences between the expected image and the actual image found in the application under test.

The **Image Verification Differences View** displays the actual and the expected image, as well as the name of the image and the compared versions of the images.

Testing Specific Environments

Silk Test Workbench supports testing several types of environments.

Apache Flex Support

Silk Test Workbench provides built-in support for testing Apache Flex applications using Internet Explorer and the Standalone Flash Player, and Adobe AIR applications built with Apache Flex 4 or later.

Silk Test Workbench also supports multiple application domains in Apache Flex 3.x and 4.x applications, which enables you to test sub-applications. Silk Test Workbench recognizes each sub-application in the locator hierarchy tree as an application tree with the relevant application domain context. At the root level in the locator attribute table, Apache Flex 4.x sub-applications use the `SparkApplication` class. Apache Flex 3.x sub-applications use the `FlexApplication` class.

Supported Controls

For a complete list of the record and playback controls available for Apache Flex testing, see the *Flex Class Reference*.

Note: The Silk Test Flex Automation SDK is based on the Automation API for Apache Flex. The Silk Test Automation SDK supports the same components in the same manner that the Automation API for Apache Flex supports them. For instance, the `typekey` statement in the Flex Automation API does not support all keys. You can use the input text statement to resolve this issue. For more information about using the Flex Automation API, see the *Apache Flex Release Notes*.

Configuring Flex Applications to Run in Adobe Flash Player

To run an Apache Flex application in Flash Player, one or both of the following must be true:

- The developer who creates the Flex application must compile the application as an EXE file. When a user launches the application, it will open in Flash Player. Install Windows Flash Player from *http://www.adobe.com/support/flashplayer/downloads.html*.
- The user must have Windows Flash Player Projector installed. When a user opens a Flex .SWF file, he can configure it to open in Flash Player. Windows Flash Projector is not installed when Flash Player is installed unless you install the Apache Flex developer suite. Install Windows Flash Projector from *http://www.adobe.com/support/flashplayer/downloads.html*.

1. For Microsoft Windows 7 and Microsoft Windows Server 2008 R2, configure Flash Player to run as administrator. Perform the following steps:
   a) Right-click the Adobe Flash Player program shortcut or the `FlashPlayer.exe` file, then click **Properties**.
   b) In the **Properties** dialog box, click the **Compatibility** tab.
   c) Check the **Run this program as an administrator** check box and then click **OK**.
2. Start the .SWF file in Flash Player from the command prompt (cmd.exe) by typing:
   `"<Application_Install_Directory>\ApplicationName.swf"`
   By default, the <*SilkTest_Install_Directory*> is located at `Program Files\Silk\Silk Test`.

Launching the Component Explorer

Silk Test provides a sample Apache Flex application, the Component Explorer. Compiled with the Adobe Automation SDK and the Silk Test specific automation implementation, the Component Explorer is pre-configured for testing.

In Internet Explorer, open *http://demo.borland.com/flex/SilkTest19.0/index.html*. The application launches in your default browser.

Testing Apache Flex Applications

Silk Test provides built-in support for testing Apache Flex applications. Silk Test also provides several sample Apache Flex applications. You can access the sample applications at *http://demo.borland.com/flex/SilkTest19.0/index.html*.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Before you can test your own Apache Flex application, your Apache Flex developers must perform the following steps:

- Enabling your Apache Flex application for testing
- Creating testable Apache Flex applications
- Coding Apache Flex containers
- Implementing automation support for custom controls

To test your own Apache Flex application, follow these steps:

- Configuring security settings for your local Flash Player
- Recording a visual test
- Playing back a visual test
- Customizing Apache Flex scripts
- Testing a custom Apache Flex control

Note: Loading an Apache Flex application and initializing the Flex automation framework may take some time depending on the machine on which you are testing and the complexity of your Apache Flex application. Set the Window timeout value to a higher value to enable your application to fully load.

Testing Apache Flex Custom Controls

Silk Test Workbench supports testing Apache Flex custom controls. However, by default, Silk Test Workbench cannot record and playback the individual sub-controls of the custom control.

For testing custom controls, the following options exist:

- Basic support

  With basic support, you use dynamic invoke to interact with the custom control during replay. Use this low-effort approach when you want to access properties and methods of the custom control in the test application that Silk Test Workbench does not expose. The developer of the custom control can also add methods and properties to the custom control specifically for making the control easier to test. A user can then call those methods or properties using the dynamic invoke feature.

  The advantages of basic support include:

  - Dynamic invoke requires no code changes in the test application.
  - Using dynamic invoke is sufficient for most testing needs.

  The disadvantages of basic support include:

- No specific class name is included in the locator, for example Silk Test Workbench records `//FlexBox` rather than `//FlexSpinner`.
- Only limited recording support.
- Silk Test Workbench cannot replay events.

For more details about dynamic invoke, including an example, see *Dynamically Invoking Apache Flex Methods*.

- Advanced support

  With advanced support, you create specific automation support for the custom control. This additional automation support provides recording support and more powerful play-back support. The advantages of advanced support include:

  - High-level recording and playback support, including the recording and replaying of events.
  - Silk Test Workbench treats the custom control exactly the same as any other built-in Apache Flex control.
  - Seamless integration into Silk Test Workbench API
  - Silk Test Workbench uses the specific class name in the locator, for example Silk Test Workbench records `//FlexSpinner`.

  The disadvantages of advanced support include:

  - Implementation effort is required. The test application must be modified and the Open Agent must be extended.

Dynamically Invoking Flex Methods

You can call methods, retrieve properties, and set properties on controls that Silk Test Workbench does not expose by using the dynamic invoke feature. This feature is useful for working with custom controls and for working with controls that Silk Test Workbench supports without customization.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

Call dynamic methods on objects with the `Invoke` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.

Note: Typically, most properties are read-only and cannot be set.

Supported Methods and Properties

The following methods and properties can be called:

- Methods and properties that Silk Test Workbench supports for the control.
- All public methods that the Flex API defines
- If the control is a custom control that is derived from a standard control, all methods and properties from the standard control can be called.

Supported Parameter Types

The following parameter types are supported:

- All built-in Silk Test Workbench types

  Silk Test Workbench types includes primitive types (such as boolean, int, string), lists, and other types (such as Point)

Returned Values

The following values are returned for properties and methods that have a return value:

- The correct value for all built-in Silk Test Workbench types. These types are listed in the *Supported Parameter Types* section.
- All methods that have no return value return `null` in C# or `Nothing` in VB.

Examples

For example, when an application developer creates a custom calculator control that offers the following methods and the following property:

```
public function reset() : void
public function add(number1 : int, number2 : int) : int
public function get description : String
```

The tester can call the methods directly from his test. For example:

```
customControl.Invoke("reset")
Dim sum as Integer = customControl.Invoke("add", 1, 2)
Dim description As String = customControl.GetProperty("description")
```

Defining a Custom Control in the Test Application

Typically, the test application already contains custom controls, which were added during development of the application. If your test application already includes custom controls, you can proceed to *Testing a Flex Custom Control Using Dynamic Invoke* or to *Testing a Custom Control Using Automation Support*.

This procedure shows how a Flex application developer can create a spinner custom control in Flex. The spinner custom control that we create in this topic is used in several topics to illustrate the process of implementing and testing a custom control.

The spinner custom control includes two buttons and a textfield, as shown in the following graphic.

The user can click **Down** to decrement the value that is displayed in the textfield and click **Up** to increment the value in the textfield.

The custom control offers a public "Value" property that can be set and retrieved.

1. In the test application, define the layout of the control.
   For example, for the spinner control type:
   ```
   <?xml version="1.0" encoding="utf-8"?>
   <customcontrols:SpinnerClass xmlns:mx="http://www.adobe.com/2006/mxml"
   xmlns:controls="mx.controls.*" xmlns:customcontrols="customcontrols.*">
       <controls:Button id="downButton" label="Down" />
       <controls:TextInput id="text" enabled="false" />
       <controls:Button id="upButton" label="Up"/>
   </customcontrols:SpinnerClass>
   ```
2. Define the implementation of the custom control.
   For example, for the spinner control type:
   ```
   package customcontrols
   {
       import flash.events.MouseEvent;

       import mx.containers.HBox;
       import mx.controls.Button;
       import mx.controls.TextInput;
       import mx.core.UIComponent;
   ```

```
    import mx.events.FlexEvent;

    [Event(name="increment",     type="customcontrols.SpinnerEvent")]
    [Event(name="decrement",     type="customcontrols.SpinnerEvent")]

    public class SpinnerClass extends HBox
    {
        public var downButton : Button;
        public var upButton : Button;
        public var text : TextInput;
        public var ssss: SpinnerAutomationDelegate;
        private var _lowerBound : int = 0;
        private var _upperBound : int = 5;

        private var _value : int = 0;
        private var _stepSize : int = 1;


        public function SpinnerClass() {
            addEventListener(FlexEvent.CREATION_COMPLETE,
creationCompleteHandler);
        }

        private function creationCompleteHandler(event:FlexEvent) : void {
            downButton.addEventListener(MouseEvent.CLICK,
downButtonClickHandler);
            upButton.addEventListener(MouseEvent.CLICK,
upButtonClickHandler);
            updateText();
        }

        private function downButtonClickHandler(event : MouseEvent) : void {
            if(Value - stepSize >= lowerBound) {
                Value = Value - stepSize;
            }
            else {
                Value = upperBound - stepSize + Value - lowerBound + 1;
            }

            var spinnerEvent : SpinnerEvent = new
SpinnerEvent(SpinnerEvent.DECREMENT);
            spinnerEvent.steps = _stepSize;
            dispatchEvent(spinnerEvent);
        }

        private function upButtonClickHandler(event : MouseEvent) : void {
            if(cValue <= upperBound - stepSize) {
                Value = Value + stepSize;
            }
            else {
                Value = lowerBound + Value + stepSize - upperBound - 1;
            }

            var spinnerEvent : SpinnerEvent = new
SpinnerEvent(SpinnerEvent.INCREMENT);
            spinnerEvent.steps = _stepSize;
            dispatchEvent(spinnerEvent);
        }

        private function updateText() : void {
            if(text != null) {
                text.text = _value.toString();
            }
        }
```

```
        public function get Value() : int {
            return _value;
        }

        public function set Value(v : int) : void {
            _value = v;
            if(v < lowerBound) {
                _value = lowerBound;
            }
            else if(v > upperBound) {
                _value = upperBound;
            }
            updateText();

        }

        public function get stepSize() : int {
            return _stepSize;
        }

        public function set stepSize(v : int) : void {
            _stepSize = v;
        }

        public function get lowerBound() : int {
            return _lowerBound;
        }

        public function set lowerBound(v : int) : void {
            _lowerBound = v;
            if(Value < lowerBound) {
                Value = lowerBound;
            }
        }

        public function get upperBound() : int {
            return _upperBound;
        }

        public function set upperBound(v : int) : void {
            _upperBound = v;
            if(Value > upperBound) {
                Value = upperBound;
            }
        }
    }
}
```

3. Define the events that the control uses.
   For example, for the spinner control type:

```
package customcontrols
{
    import flash.events.Event;

    public class SpinnerEvent extends Event
    {
        public static const INCREMENT : String = "increment";
        public static const DECREMENT : String = "decrement";

        private var _steps : int;

        public function SpinnerEvent(eventName : String) {
            super(eventName);
```

```
        }

        public function set steps(value:int) : void {
            _steps = value;
        }

        public function get steps() : int {
            return _steps;
        }

    }
}
```

The next step is to implement automation support for the test application.

Testing a Flex Custom Control Using Dynamic Invoke

Silk Test Workbench provides record and playback support for custom controls using dynamic invoke to interact with the custom control during replay. Use this low-effort approach when you want to access properties and methods of the custom control in the test application that Silk Test Workbench does not expose. The developer of the custom control can also add methods and properties to the custom control specifically for making the control easier to test.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

1. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.
2. Call dynamic methods on objects with the `Invoke` method.
3. Call multiple dynamic methods on objects with the `InvokeMethods` method.
4. Call multiple dynamic methods on objects with the `invokeMethods` method.
5. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.
6. Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method.

Example

The following example tests a spinner custom control that includes two buttons and a textfield, as shown in the following graphic.

The user can click **Down** to decrement the value that is displayed in the textfield and click **Up** to increment the value in the textfield.

The custom control offers a public "Value" property that can be set and retrieved.

To set the spinner's value to 4, type the following:

```
Dim spinner = Desktop.Find("//
FlexBox[@className=customcontrols.Spinner]")
spinner.SetProperty("Value", 4)
```

Testing a Custom Control Using Automation Support

You can create specific automation support for the custom control. This additional automation support provides recording support and more powerful play-back support. To create automation support, the test application must be modified and the Open Agent must be extended.

Before you can test a custom control in Silk Test Workbench, perform the following steps:

- Define the custom control in the test application
- Implement automation support

After the test application has been modified and includes automation support, perform the following steps:

1. For visual tests, no additional configuration is needed. You can record and playback visual tests for the custom control as needed.
2. For scripts, record the script and make manual modifications to fit the custom control.
   For example, the following code shows how to increment the spinner's value by 3 by using the "Increment" method that has been implemented in the automation delegate:

   ```
   _desktop.TestObject("//FlexSpinner[@caption='index:
   1']").Invoke("Increment", 3)
   ```

   The following example shows how to set the value of the spinner to 3.

   ```
   _desktop.TestObject("//FlexSpinner[@caption='index:
   1']").SetProperty("Value", 3)
   ```

Implementing Automation Support for a Custom Control

Before you can test a custom control, implement automation support (the automation delegate) in ActionScript for the custom control and compile that into the test application.

The following procedure uses a custom Flex spinner control to demonstrate how to implement automation support for a custom control. The spinner custom control includes two buttons and a textfield, as shown in the following graphic.

The user can click **Down** to decrement the value that is displayed in the textfield and click **Up** to increment the value in the textfield.

The custom control offers a public "Value" property that can be set and retrieved.

1. Implement automation support (the automation delegate) in ActionScript for the custom control.
   For further information about implementing an automation delegate, see the Adobe Live Documentation at *http://livedocs.adobe.com/flex/3/html/help.html?content=functest_components2_14.html*.
   In this example, the automation delegate adds support for the methods "increment", "decrement". The example code for the automation delegate looks like this:

   ```
   package customcontrols
   {
       import flash.display.DisplayObject;
       import mx.automation.Automation;
       import customcontrols.SpinnerEvent;
       import mx.automation.delegates.containers.BoxAutomationImpl;
       import flash.events.Event;
       import mx.automation.IAutomationObjectHelper;
       import mx.events.FlexEvent;
       import flash.events.IEventDispatcher;
       import mx.preloaders.DownloadProgressBar;
       import flash.events.MouseEvent;
   ```

```
    import mx.core.EventPriority;

    [Mixin]
    public class SpinnerAutomationDelegate extends BoxAutomationImpl
    {

        public static function init(root:DisplayObject) : void {
            // register delegate for the automation
            Automation.registerDelegateClass(Spinner,
SpinnerAutomationDelegate);
        }

        public function SpinnerAutomationDelegate(obj:Spinner) {
            super(obj);
            // listen to the events of interest (for recording)
            obj.addEventListener(SpinnerEvent.DECREMENT, decrementHandler);
            obj.addEventListener(SpinnerEvent.INCREMENT, incrementHandler);
        }

        protected function decrementHandler(event : SpinnerEvent) : void {
            recordAutomatableEvent(event);
        }

        protected function incrementHandler(event : SpinnerEvent) : void {
            recordAutomatableEvent(event);
        }

        protected function get spinner() : Spinner {
            return uiComponent as Spinner;
        }

        //----------------------------------
        //  override functions
        //----------------------------------

        override public function get automationValue():Array {
            return [ spinner.Value.toString() ];
        }

        private function replayClicks(button : IEventDispatcher, steps :
int) : Boolean {
            var helper : IAutomationObjectHelper =
Automation.automationObjectHelper;
            var result : Boolean;
            for(var i:int; i < steps; i++) {
                helper.replayClick(button);
            }
            return result;
        }

        override public function
replayAutomatableEvent(event:Event):Boolean {

            if(event is SpinnerEvent) {
                var spinnerEvent : SpinnerEvent = event as SpinnerEvent;
                if(event.type == SpinnerEvent.INCREMENT) {
                    return replayClicks(spinner.upButton,
spinnerEvent.steps);
                }
                else if(event.type == SpinnerEvent.DECREMENT) {
                    return replayClicks(spinner.downButton,
spinnerEvent.steps);
                }
                else {
```

```
                    return false;
                }

            }
            else {
                return super.replayAutomatableEvent(event);
            }
        }

        // do not expose the child controls (i.e the buttons and the
textfield) as individual controls
        override public function get numAutomationChildren():int {
            return 0;
        }

    }
}
```

2. To introduce the automation delegate to the Open Agent, create an XML file that describes the custom control.

   The class definition file contains information about all instrumented Flex components. This file (or files) provides information about the components that can send events during recording and accept events for replay. The class definition file also includes the definitions for the supported properties.

   The XML file for the spinner custom control looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<TypeInformation>
    <ClassInfo Name="FlexSpinner" Extends="FlexBox">
        <Implementation
            Class="customcontrols.Spinner" />
        <Events>
            <Event Name="Decrement">
                <Implementation
                    Class="customcontrols.SpinnerEvent"
                    Type="decrement" />
                <Property Name="steps">
                    <PropertyType Type="integer" />
                </Property>
            </Event>
            <Event Name="Increment">
                <Implementation
                    Class="customcontrols.SpinnerEvent"
                    Type="increment" />
                <Property Name="steps">
                    <PropertyType Type="integer" />
                </Property>
            </Event>
        </Events>
        <Properties>
            <Property Name="lowerBound" accessType="read">
                <PropertyType Type="integer" />
            </Property>
            <Property Name="upperBound" accessType="read">
                <PropertyType Type="integer" />
            </Property>
            <!-- expose read and write access for the Value property -->
            <Property Name="Value" accessType="both">
                <PropertyType Type="integer" />
            </Property>
            <Property Name="stepSize" accessType="read">
                <PropertyType Type="integer" />
            </Property>
        </Properties>
```

```
    </ClassInfo>
</TypeInformation>
```

3. Include the XML file for the custom control in the folder that includes all the XML files that describe all classes and their methods and properties for the supported Flex controls.

   Silk Test contains several XML files that describe all classes and their methods and properties for the supported Flex controls. Those XML files are located in the <<Silk Test_install_directory>\ng\agent\plugins\com.borland.fastxd.techdomain.flex.agent_<version>\config\automationEnvironment folder.

   If you provide your own XML file, you must copy your XML file into this folder. When the Open Agent starts and initializes support for Apache Flex, it reads the contents of this directory.

   To test the Flex Spinner sample control, you must copy the CustomControls.xml file into this folder. If the Open Agent is currently running, restart it after you copy the file into the folder.

You can now record and playback visual tests or scripts that include the custom control.

Flex Class Definition File

The class definition file contains information about all instrumented Flex components. This file (or files) provides information about the components that can send events during recording and accept events for replay. The class definition file also includes the definitions for the supported properties.

Silk Test contains several XML files that describe all classes/events/properties for the common Flex common and specialized controls. Those XML files are located in the <Silk Test_install_directory>\ng\agent\plugins\com.borland.fastxd.techdomain.flex.agent_<*version*>\config\automationEnvironment folder.

If you provide your own XML file, you must copy your XML file into this folder. When the Silk Test agent starts and initializes support for Apache Flex, it reads the contents of this directory.

The XML file has the following basic structure:

```
<TypeInformation>

<ClassInfo>

<Implementation />

<Events>

<Event />

…

</Events>

<Properties>

<Property />

…

</Properties>

</ClassInfo>

</TypeInformation>
```

Customizing Apache Flex Scripts

You can manually customize your Flex scripts. You can insert verifications manually using the `Verify` function on Flex object properties. Each Flex object has a list of properties that you can verify. For a list of the properties available for verification, review the *Flex Class Reference*.

1. Record a test for your Flex application.
2. Open the script file that you want to customize.
3. Manually type the code that you want to add.

Testing Multiple Flex Applications on the Same Web Page

When multiple Flex applications exist on the same Web page, Silk Test Workbench uses the Flex application ID or the application size property to determine which application to test. If multiple applications exist on the same page, but they are different sizes, Silk Test Workbench uses the size property to determine on which application to perform any actions and no additional steps are necessary.

Silk Test Workbench uses JavaScript to find the Flex application ID to determine on which application to perform any actions if:

- Multiple Flex applications exist on a single Web page
- Those applications are the same size

Note: In this situation, if JavaScript is not enabled on the browser machine, an error occurs when a script runs.

1. Enable JavaScript.
2. In Internet Explorer, perform the following steps:
   a) Choose **Tools** > **Internet Options**.
   b) Click the **Security** tab.
   c) Click **Custom level**.
   d) In the **Scripting** section, under **Active Scripting**, click **Enable** and click **OK**.
3. Follow the steps in *Testing Apache Flex Applications*.

Note: If a frame exists on the Web page and the applications are the same size, this method will not work.

Adobe AIR Support

Silk Test Workbench supports testing with Adobe AIR for applications that are compiled with the Flex 4 compiler. For details about supported versions, check the *Release Notes* for the latest information.

Silk Test provides a sample Adobe AIR application. You can access the sample application at *http://demo.borland.com/flex/SilkTest19.0/index.html* and then click the Adobe AIR application that you want to use. You can select the application with or without automation. In order to execute the AIR application, you must install the Adobe AIR Runtime.

Overview of the Flex Select Method Using Name or Index

You can record Flex `Select` methods using the `Name` or `Index` of the control that you select. By default, Silk Test Workbench records `Select` methods using the name of the control. However, you can change

your environment to record `Select` events using the index for the control, or you can switch between the name and index for recording.

You can record `Select` events using the index for the following controls:

- `FlexList`
- `FlexTree`
- `FlexDataGrid`
- `FlexAdvancedDataGrid`
- `FlexOLAPDataGrid`
- `FlexComboBox`

The default setting is ItemBasedSelection (Select event), which uses the name control. To use the index, you must adapt the AutomationEnvironment to use the IndexBasedSelection (SelectIndex event). To change the behavior for one of these classes, you must modify the FlexCommonControls.xml, AdvancedDataGrid.xml, or OLAPDataGrid.xml file using the following code. Those XML files are located in the `<Silk Test_install_directory>\ng\agent\plugins \com.borland.fastxd.techdomain.flex.agent_< version>\config \automationEnvironment` folder. Make the following adaptations in the corresponding xml file.

```
<ClassInfo Extends="FlexList" Name="FlexControlName"
EnableIndexBasedSelection=”true” >

…

</ClassInfo>
```

With this adaption the IndexBasedSelection is used for recording `FlexList::SelectIndex` events. Setting the `EnableIndexBasedSelection=` to `false` in the code or removing the Boolean returns recording to using the name (`FlexList::Select` events).

Note: You must re-start your application, which automatically re-starts the Silk Test Agent, in order for these changes to become active.

Selecting an Item in the FlexDataGrid Control

Select an item in the FlexDataGrid control using the index value or the content value.

1. To select an item in the FlexDataGrid control using the index value, use the `SelectIndex` method. For example, type `FlexDataGrid.SelectIndex(1)`.
2. To select an item in the FlexDataGrid control using the content value, use the `Select` method.

   Identify the row that you want to select with the required formatted string. Items must be separated by a pipe (" | "). At least one Item must be enclosed by two stars ("*"). This identifies the item where the click will be performed.

   The syntax is: `FlexDataGrid.Select("*Item1* | Item2 | Item3")`

Enabling Your Flex Application for Testing

To enable your Flex application for testing, your Apache Flex developers must include the following components in the Flex application:

- Apache Flex Automation Package
- Silk Test Automation Package

Apache Flex Automation Package

The Flex automation package provides developers with the ability to create Flex applications that use the Automation API. You can download the Flex automation package from Adobe's website, *http://www.adobe.com*. The package includes:

- Automation libraries – the automation.swc and automation_agent.swc libraries are the implementations of the delegates for the Flex framework components. The automation_agent.swc file and its associated resource bundle are the generic agent mechanism. An agent, such as the Silk Test Agent, builds on top of these libraries.
- Samples

Note: The Silk Test Flex Automation SDK is based on the Automation API for Flex. The Silk Test Automation SDK supports the same components in the same manner that the Automation API for Flex supports them. For instance, the `typekey` statement in the Flex Automation API does not support all keys. You can use the input text statement to resolve this issue. For more information about using the Flex Automation API, see the *Apache Flex Release Notes*.

Silk Test Automation Package

Silk Test's Open Agent uses the Apache Flex automation agent libraries. The FlexTechDomain.swc file contains the Silk Test specific implementation.

You can enable your application for testing using either of the following methods:

- Linking automation packages to your Flex application
- Run-time loading

Linking Automation Packages to Your Flex Application

You must precompile Flex applications that you plan to test. The functional testing classes are embedded in the application at compile time, and the application has no external dependencies for automated testing at run time.

When you embed functional testing classes in your application SWF file at compile time, the size of the SWF file increases. If the size of the SWF file is not important, use the same SWF file for functional testing and deployment. If the size of the SWF file is important, generate two SWF files, one with functional testing classes embedded and one without. Use the SWF file that does not include the embedded testing classes for deployment.

When you precompile the Flex application for testing, in the include-libraries compiler option, reference the following files:

- automation.swc
- automation_agent.swc
- FlexTechDomain.swc
- automation_charts.swc (include only if your application uses charts and Flex 2.0)
- automation_dmv.swc (include if your application uses charts and Flex > 3.x)
- automation_flasflexkit.swc (include if your application uses embedded flash content)
- automation_spark.swc (include if your application uses the new Flex 4.x controls)
- automation_air.swc (include if your application is an AIR application)
- automation_airspark.swc (include if your application is an AIR application and uses new Flex 4.x controls)

When you create the final release version of your Flex application, you recompile the application without the references to these SWC files. For more information about using the automation SWC files, see the *Apache Flex Release Notes*.

If you do not deploy your application to a server, but instead request it by using the file protocol or run it from within Apache Flex Builder, you must include each SWF file in the local-trusted sandbox. This requires

additional configuration information. Add the additional configuration information by modifying the compiler's configuration file or using a command-line option.

Note: The Silk Test Flex Automation SDK is based on the Automation API for Flex. The Silk Test Automation SDK supports the same components in the same manner that the Automation API for Flex supports them. For instance, when an application is compiled with automation code and successive SWF files are loaded, a memory leak occurs and the application runs out of memory eventually. The Flex Control Explorer sample application is affected by this issue. The workaround is to not compile the application SWF files that Explorer loads with automation libraries. For example, compile only the Explorer main application with automation libraries. Another alternative is to use the module loader instead of swfloader. For more information about using the Flex Automation API, see the *Apache FlexRelease Notes*.

Precompiling the Flex Application for Testing

You can enable your application for testing by precompiling your application for testing or by using run-time loading.

1. Include the automation.swc, automation_agent.swc, and FlexTechDomain.swc libraries in the compiler's configuration file by adding the following code to the configuration file:

```
<include-libraries>

...

<library>/libs/automation.swc</library>

<library>/libs/automation_agent.swc</library>

<library>pathinfo/FlexTechDomain.swc</library>

</include-libraries>
```

   Note: If your application uses charts, you must also add the automation_charts.swc file.

2. Specify the location of the automation.swc, automation_agent.swc, and FlexTechDomain.swc libraries using the include-libraries compiler option with the command-line compiler.

   The configuration files are located at:

   Apache Flex 2 SDK – <flex_installation_directory>/frameworks/flex-config.xml

   Apache Flex Data Services – <flex_installation_directory>/flex/WEB-INF/flex/flex-config.xml

   The following example adds the automation.swc and automation_agent.swc files to the application:

```
mxmlc -include-libraries+=../frameworks/libs/automation.swc;../frameworks/
libs/
automation_agent.swc;pathinfo/FlexTechDomain.swc MyApp.mxml
```

   Note: Explicitly setting the include-libraries option on the command line overwrites, rather than appends, the existing libraries. If you add the automation.swc and automation_agent.swc files using the include-libraries option on the command line, ensure that you use the += operator. This appends rather than overwrites the existing libraries that are included.

   Note: The Silk Test Flex Automation SDK is based on the Automation API for Flex. The Silk Test Automation SDK supports the same components in the same manner that the Automation API for Flex supports them. For instance, when an application is compiled with automation code and successive SWF files are loaded, a memory leak occurs and the application runs out of memory eventually. The Flex Control Explorer sample application is affected by this issue. The workaround is to not compile the application SWF files that Explorer loads with automation libraries. For example, compile only the Explorer main application with automation libraries. Another alternative

is to use the module loader instead of swfloader. For more information about using the Flex Automation API, see the *Apache FlexRelease Notes*.

Run-Time Loading

You can load Flex automation support at run time using the Silk Test Flex Automation Launcher. This application is compiled with the automation libraries and loads your application with the SWFLoader class. This automatically enables your application for testing without compiling automation libraries into your SWF file. The Silk Test Flex Automation Launcher is available in HTML and SWF file formats.

Limitations

- The Flex Automation Launcher Application automatically becomes the root application. If your application must be the root application, you cannot load automation support with the Silk Test Flex Automation Launcher.
- Testing applications that load external libraries – Applications that load other SWF file libraries require a special setting for automated testing. A library that is loaded at run time (including run-time shared libraries (RSLs) must be loaded into the ApplicationDomain of the loading application. If the SWF file used in the application is loaded in a different application domain, automated testing record and playback will not function properly. The following example shows a library that is loaded into the same ApplicationDomain:

```
import flash.display.*;

import flash.net.URLRequest;

import flash.system.ApplicationDomain;

import flash.system.LoaderContext;


var ldr:Loader = new Loader();


var urlReq:URLRequest = new URLRequest("RuntimeClasses.swf");

var context:LoaderContext = new LoaderContext();

context.applicationDomain = ApplicationDomain.currentDomain;

loader.load(request, context);
```

Loading at Run-Time

1. Copy the content of the `Silk\Silk Test\ng\AutomationSDK\Flex\<version>\FlexAutomationLauncher` directory into the directory of the Flex application that you are testing.
2. Open `FlexAutomationLauncher.html` in Windows Explorer and add the following parameter as a suffix to the file path:

   ```
   ?automationurl=YourApplication.swf
   ```

   where *YourApplication.swf* is the name of the SWF file for your Flex application.
3. Add `file:///` as a prefix to the file path.
   For example, if your file URL includes a parameter, such as: `?automationurl=explorer.swf`, type: .

   ```
   file:///C:/Program%20Files/Silk/Silk Test/ng/sampleapplications/Flex/3.2/
   FlexControlExplorer32/FlexAutomationLauncher.html?automationurl=explorer.swf
   ```

Using the Command Line to Add Configuration Information

To specify the location of the automation.swc, automation_agent.swc, and FlexTechDomain.swc libraries using the command-line compiler, use the include-libraries compiler option.

The following example adds the `automation.swc` and `automation_agent.swc` files to the application:

```
mxmlc -include-libraries+=../frameworks/libs/automation.swc;../frameworks/
libs/
automation_agent.swc;pathinfo/FlexTechDomain.swc MyApp.mxml
```

Note: If your application uses charts, you must also add the `automation_charts.swc` file to the include-libraries compiler option.

Explicitly setting the include-libraries option on the command line overwrites, rather than appends, the existing libraries. If you add the `automation.swc` and `automation_agent.swc` files using the include-libraries option on the command line, ensure that you use the += operator. This appends rather than overwrites the existing libraries that are included.

To add automated testing support to a Flex Builder project, you must also add the `automation.swc` and `automation_agent.swc` files to the include-libraries compiler option.

Passing Parameters into a Flex Application

You can pass parameters into a Flex application using the following procedures.

Passing Parameters into a Flex Application Before Runtime

You can pass parameters into a Flex application before runtime using automation libraries.

1. Compile your application with the appropriate automation libraries.
2. Use the standard Flex mechanism for the parameter as you typically would.

Passing Parameters into a Flex Application at Runtime Using the Flex Automation Launcher

Before you begin this task, prepare your application for run-time loading.

1. Open the `FlexAutomationLauncher.html` file or create a file using `FlexAutomationLauncher.html` as an example.
2. Navigate to the following section:

```
<script language="JavaScript" type="text/javascript">
                AC_FL_RunContent(eef
                   "src", "FlexAutomationLauncher",
                   "width", "100%",
                   "height", "100%",
                   "align", "middle",
                   "id", "FlexAutomationLauncher",
                   "quality", "high",
                   "bgcolor", "white",
                   "name", "FlexAutomationLauncher",
                   "allowScriptAccess","sameDomain",
```

```
                    "type", "application/x-shockwave-flash",

                    "pluginspage", "http://www.adobe.com/go/getflashplayer",

                    "flashvars", "yourParameter=yourParameterValue"+
"&automationurl=YourApplication.swf"



                );

        </script>
```

Note: Do not change the "`FlexAutomationLauncher`" value for "`src`", "`id`", or "`name`."

3. Add your own parameter to "*yourParameter=yourParameterValue*".
4. Pass the name of the Flex application that you want to test as value for the "*&automationurl=YourApplication.swf*" value.
5. Save the file.

Creating Testable Flex Applications

As a Flex developer, you can employ techniques to make Flex applications as "test friendly" as possible. These include:

- Providing Meaningful Identification of Objects
- Avoiding Duplication of Objects

Providing Meaningful Identification of Objects

To create "test friendly" applications, ensure that objects are identifiable in scripts. You can set the value of the ID property for all controls that are tested, and ensure that you use a meaningful string for that ID property.

To provide meaningful identification of objects:

- Give all testable MXML components an ID to ensure that the test script has a unique identifier to use when referring to that Flex control.
- Make these identifiers as human-readable as possible to make it easier for the user to identify that object in the testing script. For example, set the id property of a Panel container inside a TabNavigator to `submit_panel` rather than `panel1` or `p1`.

When working with Silk Test Workbench, an object is automatically given a name depending on certain tags, for instance, id, childIndex. If there is no value for the id property, Silk Test Workbench uses other properties, such as the childIndex property. Assigning a value to the id property makes the testing scripts easier to read.

Avoiding Duplication of Objects

Automation agents rely on the fact that some properties of object instances will not be changed during run time. If you change the Flex component property that is used by Silk Test Workbench as the object name at run time, unexpected results can occur. For example, if you create a Button control without an `automationName` property, and you do not initially set the value of its label property, and then later set the value of the `label` property, problems might occur. In this case, Silk Test Workbench uses the value of the label property of Button controls to identify an object if the automationName property is not set. If you later set the value of the `label` property, or change the value of an existing label, Silk Test Workbench identifies the object as a new object and does not reference the existing object.

To avoid duplicating objects:

- Understand what properties are used to identify objects in the agent and avoid changing those properties at run time.
- Set unique, human-readable id or `automationName` properties for all objects that are included in the recorded script.

Custom Attributes for Apache Flex Applications

Apache Flex applications use the predefined property `automationName` to specify a stable identifier for the Apache Flex control as follows:

```
<?xml version="1.0" encoding="utf-8"?>
  <s:Group xmlns:fx="http://ns.adobe.com/mxml/2009"
      xmlns:s="library://ns.adobe.com/flex/spark"
      xmlns:mx="library://ns.adobe.com/flex/mx" width="400" height="300">
    <fx:Script>
    …
    </fx:Script>
    <s:Button x="247" y="81" label="Button" id="button1" enabled="true"
click="button1_clickHandler(event)"
        automationName="AID_buttonRepeat"/>
    <s:Label x="128" y="123" width="315" height="18" id="label1"
verticalAlign="middle"
        text="awaiting your click" textAlign="center"/>
  </s:Group>
```

Apache Flex application locators look like the following:

```
…//SparkApplication//SparkButton[@caption='AID_buttonRepeat'
```

Attention: For Apache Flex applications, the *automationName* is always mapped to the locator attribute `caption` in Silk Test Workbench. If the *automationName* attribute is not specified, Silk Test Workbench maps the property `ID` to the locator attribute `caption`.

Flex AutomationName and AutomationIndex Properties

The Flex Automation API introduces the `automationName` and `automationIndex` properties. If you provide the `automationName`, Silk Test Workbench uses this value for the recorded window declaration's name. Providing a meaningful name makes it easier for Silk Test Workbench to identify that object. As a best practice, set the value of the `automationName` property for all objects that are part of the application's test.

Use the `automationIndex` property to assign a unique index value to an object. For instance, if two objects share the same name, assign an index value to distinguish between the two objects.

Note: The Silk Test Flex Automation SDK is based on the Automation API for Flex. The Silk Test Automation SDK supports the same components in the same manner that the Automation API for Flex supports them. For instance, when an application is compiled with automation code and successive SWF files are loaded, a memory leak occurs and the application runs out of memory eventually. The Flex Control Explorer sample application is affected by this issue. The workaround is to not compile the application SWF files that Explorer loads with automation libraries. For example, compile only the Explorer main application with automation libraries. Another alternative is to use the module loader instead of swfloader. For more information about using the Flex Automation API, see the *Apache Flex Release Notes*.

Flex Class Definition File

The class definition file contains information about all instrumented Flex components. This file (or files) provides information about the components that can send events during recording and accept events for replay. The class definition file also includes the definitions for the supported properties.

Silk Test contains several XML files that describe all classes/events/properties for the common Flex common and specialized controls. Those XML files are located in the `<Silk`

Test_install_directory>\ng\agent\plugins
\com.borland.fastxd.techdomain.flex.agent_<*version*>\config
\automationEnvironment folder.

If you provide your own XML file, you must copy your XML file into this folder. When the Silk Test agent starts and initializes support for Apache Flex, it reads the contents of this directory.

The XML file has the following basic structure:

```
<TypeInformation>

<ClassInfo>

<Implementation />

<Events>

<Event />

…

</Events>

<Properties>

<Property />

…

</Properties>

</ClassInfo>

</TypeInformation>
```

Setting the Flex automationName Property

The `automationName` property defines the name of a component as it appears in tests. The default value of this property varies depending on the type of component. For example, the `automationName` for a Button control is the label of the Button control. Sometimes, the `automationName` is the same as the `id` property for the control, but this is not always the case.

For some components, Flex sets the value of the `automationName` property to a recognizable attribute of that component. This helps testers recognize the component in their tests. Because testers typically do not have access to the underlying source code of the application, having a control's visible property define that control can be useful. For example, a Button labeled "Process Form Now" appears in the test as `FlexButton("Process Form Now")`.

If you implement a new component, or derive from an existing component, you might want to override the default value of the `automationName` property. For example, UIComponent sets the value of the `automationName` to the component's `id` property by default. However, some components use their own methods for setting the value. For example, in the Flex Store sample application, containers are used to create the product thumbnails. A container's default `automationName` would not be very useful because it is the same as the container's `id` property. So, in Flex Store, the custom component that generates a product thumbnail explicitly sets the `automationName` to the product name to make testing the application easier.

Example

The following example from the CatalogPanel.mxml custom component sets the value of the `automationName` property to the name of the item as it appears in the catalog. This is more recognizable than the default automation name.

```
thumbs[i].automationName = catalog[i].name;
```

Example

The following example sets the `automationName` property of the ComboBox control to "Credit Card List"; rather than using the `id` property, the testing tool typically uses "Credit Card List" to identify the ComboBox in its scripts:

```
<?xml version="1.0"?>
<!-- at/SimpleComboBox.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      [Bindable]
      public var cards: Array = [
        {label:"Visa", data:1},
        {label:"MasterCard", data:2},
        {label:"American Express", data:3}
      ];

      [Bindable]
      public var selectedItem:Object;
      ]
   ]>
  </mx:Script>
  <mx:Panel title="ComboBox Control Example">
    <mx:ComboBox id="cb1" dataProvider="{cards}"
      width="150"
      close="selectedItem=ComboBox(event.target).selectedItem"
      automationName="Credit Card List"
    />
    <mx:VBox width="250">
      <mx:Text width="200" color="blue" text="Select a type of
credit card." />
      <mx:Label text="You selected: {selectedItem.label}"/>
      <mx:Label text="Data: {selectedItem.data}"/>
    </mx:VBox>
  </mx:Panel>
</mx:Application>
```

Setting the value of the `automationName` property ensures that the object name will not change at run time. This helps to eliminate unexpected results.

If you set the value of the `automationName` property, tests use that value rather than the default value. For example, by default, Silk Test Workbench uses a Button control's label property as the name of the Button in the script. If the label changes, the script can break. You can prevent this from happening by explicitly setting the value of the `automationName` property.

Buttons that have no label, but have an icon, are recorded by their index number. In this case, ensure that you set the `automationName` property to something meaningful so that the tester can recognize the Button in the script. After the value of the `automationName` property is set, do not change the value during the component's life cycle. For item renderers, use the `automationValue` property rather than the `automationName` property. To use the `automationValue` property, override the

`createAutomationIDPart()` method and return a new value that you assign to the `automationName` property, as the following example shows:

```
<mx:List xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      import mx.automation.IAutomationObject;
      override public function
      createAutomationIDPart(item:IAutomationObject):Object {
        var id:Object = super.createAutomationIDPart(item);
        id["automationName"] = id["automationIndex"];
        return id;
      }
    ]]>
  </mx:Script>
</mx:List>
```

Use this technique to add index values to the children of any container or list-like control. There is no method for a child to specify an index for itself.

Setting the Flex Select Method to Use Name or Index

You can record Flex `Select` methods using the `Name` or `Index` of the control that you select. By default, Silk Test records `Select` methods using the name of the control. However, you can change your environment to record `Select` events using the index for the control, or you can switch between the name and index for recording.

1. Determine which class you want to modify to use the Index.

   You can record `Select` events using the index for the following controls:

   - `FlexList`
   - `FlexTree`
   - `FlexDataGrid`
   - `FlexOLAPDataGrid`
   - `FlexComboBox`
   - `FlexAdvancedDataGrid`

2. Determine which XML file is related to the class that you want to modify.

   The XML files related to the preceding controls include: `FlexCommonControls.xml`, `AdvancedDataGrid.xml`, or `OLAPDataGrid.xml`.

3. Navigate to the XML files that are related to the class that you want to modify.

   The XML files are located in the `<Silk Test_install_directory>\ng\agent\plugins\com.borland.fastxd.techdomain.flex.agent_<version>\config\automationEnvironment` folder.

4. Make the following adaptations in the corresponding XML file.

   ```
   <ClassInfo Extends="FlexList" Name="FlexControlName"
   EnableIndexBasedSelection=”true” >

   …

   </ClassInfo>
   ```

   For instance, you might use "`FlexList`" as the " `FlexControlName`" and modify the `FlexCommonControls.xml` file.

   With this adaption the IndexBasedSelection is used for recording `FlexList::SelectIndex` events.

Note: Setting the `EnableIndexBasedSelection=` to `false` in the code or removing the boolean returns recording to using the name (`FlexList::Select` events).

5. Re-start your Flex application and the Open Agent in order for these changes to become active.

Coding Flex Containers

Containers differ from other kinds of controls because they are used both to record user interactions (such as when a user moves to the next pane in an Accordion container) and to provide unique locations for controls in the testing scripts.

Adding and Removing Containers from the Automation Hierarchy

In general, the automated testing feature reduces the amount of detail about nested containers in its scripts. It removes containers that have no impact on the results of the test or on the identification of the controls from the script. This applies to containers that are used exclusively for layout, such as the HBox, VBox, and Canvas containers, except when they are being used in multiple-view navigator containers, such as the ViewStack, TabNavigator, or Accordion containers. In these cases, they are added to the automation hierarchy to provide navigation.

Many composite components use containers, such as Canvas or VBox, to organize their children. These containers do not have any visible impact on the application. As a result, you typically exclude these containers from testing because there is no user interaction and no visual need for their operations to be recordable. By excluding a container from testing, the related test script is less cluttered and easier to read.

To exclude a container from being recorded (but not exclude its children), set the container's `showInAutomationHierarchy` property to false. This property is defined by the `UIComponent` class, so all containers that are a subclass of `UIComponent` have this property. Children of containers that are not visible in the hierarchy appear as children of the next highest visible parent.

The default value of the `showInAutomationHierarchy` property depends on the type of container. For containers such as Panel, Accordion, Application, DividedBox, and Form, the default value is `true`; for other containers, such as Canvas, HBox, VBox, and FormItem, the default value is `false`.

The following example forces the VBox containers to be included in the test script's hierarchy:

```
<?xml version="1.0"?>
<!-- at/NestedButton.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
<mx:Panel title="ComboBox Control Example">
<mx:HBox id="hb">
<mx:VBox id="vb1" showInAutomationHierarchy="true">
<mx:Canvas id="c1">
<mx:Button id="b1" automationName="Nested Button 1" label="Click Me" />
</mx:Canvas>
</mx:VBox>
<mx:VBox id="vb2" showInAutomationHierarchy="true">
<mx:Canvas id="c2">
<mx:Button id="b2" automationName="Nested Button 2" label="Click Me 2" />
</mx:Canvas>
</mx:VBox>
</mx:HBox>
</mx:Panel>
</mx:Application>
```

Multiview Containers

Avoid using the same label on multiple tabs in multiview containers, such as the TabNavigator and Accordion containers. Although it is possible to use the same labels, this is generally not an acceptable UI design practice and can cause problems with control identification in your testing environment.

Flex Automation Testing Workflow

The Silk Test Workbench workflow for testing Flex applications includes:

- Automated Testing Initialization
- Automated Testing Recording
- Automated Testing Playback

Flex Automated Testing Initialization

When the user launches the Flex application, the following initialization events occur:

1. The automation initialization code associates component delegate classes with component classes.
2. The component delegate classes implement the IAutomationObject interface.
3. An instance for the AutomationManager is created in the mixin `init()` method. (The AutomationManager is a mixin.)
4. The SystemManager initializes the application. Component instances and their corresponding delegate instances are created. Delegate instances add event listeners for events of interest.
5. The Silk Test Workbench FlexTechDomain is a mixin. In the FlexTechDomain `init()` method, the FlexTechDomain registers for the `SystemManager.APPLICATION_COMPLETE` event. When the event is received, it creates a FlexTechDomain instance.
6. The FlexTechDomain instance connects via a TCP/IP socket to the Silk Test Agent on the same machine that registers for record/playback functionality.
7. The FlexTechDomain requests information about the automation environment. This information is stored in XML files and is forwarded from the Silk Test Agent to the FlexTechDomain.

Flex Automated Testing Recording

When the user records a new test in Silk Test Workbench for a Flex application, the following events occur:

1. Silk Test Workbench calls the Silk Test Agent to start recording. The Agent forwards this command to the FlexTechDomain instance.
2. FlexTechDomain notifies AutomationManager to start recording by calling `beginRecording()`. The AutomationManager adds a listener for the AutomationRecordEvent.RECORD event from the SystemManager.
3. The user interacts with the application. For example, suppose the user clicks a Button control.
4. The `ButtonDelegate.clickEventHandler()` method dispatches an AutomationRecordEvent event with the click event and Button instance as properties.
5. The AutomationManager record event handler determines which properties of the click event to store based on the XML environment information. It converts the values into proper type or format. It dispatches the record event.
6. The FlexTechDomain event handler receives the event. It calls the `AutomationManager.createID()` method to create the AutomationID object of the button. This object provides a structure for object identification. The AutomationID structure is an array of AutomationIDParts. An AutomationIDPart is created by using IAutomationObject. (The UIComponent.id, automationName, automationValue, childIndex, and label properties of the Button control are read and stored in the object. The label property is used because the XML information specifies that this property can be used for identification for the Button.)
7. FlexTechDomain uses the `AutomationManager.getParent()` method to get the logical parent of Button. The AutomationIDPart objects of parent controls are collected at each level up to the application level.
8. All the AutomationIDParts are included as part of the AutomationID object.
9. The FlexTechDomain sends the information in a call to Silk Test Workbench.
10. When the user stops recording, the `FlexTechDomain.endRecording()` method is called.

Flex Automated Testing Playback

When the user clicks the **Playback** button in Silk Test Workbench, the following events occur:

1. For each script call, Silk Test Workbench contacts the Silk Test Agent and sends the information for the script call to be executed. This information includes the complete window declaration, the event name, and parameters.
2. The Silk Test Agent forwards that information to the FlexTechDomain.
3. The FlexTechDomain uses `AutomaionManager.resolveIDToSingleObject` with the window declaration information. The AutomationManager returns the resolved object based on the descriptive information (automationName, automationIndex, id, and so on).
4. Once the Flex control is resolved, FlexTechDomain calls `AutomationManager.replayAutomatableEvent()` to replay the event.
5. The `AutomationManager.replayAutomatableEvent()` method invokes the `IAutomationObject.replayAutomatableEvent()` method on the delegate class. The delegate uses the `IAutomationObjectHelper.replayMouseEvent()` method (or one of the other replay methods, such as `replayKeyboardEvent()`) to play back the event.
6. If there are verifications in your script, FlexTechDomain invokes `AutomationManager.getProperties()` to access the values that must be verified.

Styles in Apache Flex Applications

For applications developed in Apache Flex 3.x, Silk Test Workbench does not distinguish between styles and properties. As a result, styles are exposed as properties. However, with Apache Flex 4.x, all new Flex controls, which are prefixed with `Spark`, such as `SparkButton`, do not expose styles as properties. As a result, the `GetProperty()` and `GetPropertyList()` methods for Flex 4.x controls do not return styles, such as `color` or `fontSize`, but only properties, such as `text` and `name`.

The `GetStyle(string styleName)` method returns values of styles as a string. To find out which styles exist, refer to the Adobe help located at: *http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/package-detail.html*.

If the style is not set, a `StyleNotSetException` occurs during playback.

For the Flex 3.x controls, such as `FlexTree`, you can use `GetProperty()` to retrieve styles. Or, you can use `GetStyle()`. Both the `GetProperty()` and `GetStyle()` methods work with Flex 3.x controls.

Calculating the Color Style

In Flex, the color is represented as number. It can be calculated using the following formula:

```
red*65536 + green*256 + blue
```

Example

In the following example, the script verifies whether the buttonBar in the Spark application uses font size 12.

```
 Imports SilkTest.Ntf.Flex

Public Module Main
    Dim _desktop As Desktop = Agent.Desktop

    Public Sub Main()
        Dim Application As SparkApplication
        Dim ButtonBar As SparkButtonBar
        Application = _desktop.Find( "/BrowserApplication//
BrowserWindow//
            SparkApplication" )
        ButtonBar = Application.SparkButtonBar()
```

```
        Workbench.Verify(ButtonBar.GetStyle( "fontSize" ),
"12" )
    End Sub
End Module
```

Configuring Flex Applications for Adobe Flash Player Security Restrictions

The security model in Adobe Flash Player 10 has changed from earlier versions. When you record tests that use Flash Player, recording works as expected. However, when you play back tests, unexpected results occur when high-level clicks are used in certain situations. For instance, a **File Reference** dialog box cannot be opened programmatically and when you attempt to play back this scenario, the test fails because of security restrictions.

To work around the security restrictions, you can perform a low-level click on the button that opens the dialog box. To create a low-level click, add a parameter to the `Click` method.

For example, instead of using `SparkButton::Click()`, use `SparkButton::Click(MouseButton.Left)`. A `Click()` without parameters is a high-level click and a click with parameters (such as the button) is replayed as a low-level click.

1. Record the steps that use Flash Player.
2. Navigate to the `Click` method and add a parameter.
   For example, to open the **Open File** dialog box, specify:
   ```
   SparkButton("@caption='Open File Dialog…'").Click(MouseButton.Left)
   ```
   When you play back the test, it works as expected.

Attributes for Apache Flex Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Flex applications include:

- automationName
- caption (similar to automationName)
- automationClassName (e.g. `FlexButton`)
- className (the full qualified name of the implementation class, e.g. `mx.controls.Button`)
- automationIndex (the index of the control in the view of the FlexAutomation, e.g. `index:1`)
- index (similar to automationIndex but without the prefix, e.g. `1`)
- id (the id of the control)
- windowId (similar to id)
- label (the label of the control)
- All dynamic locator attributes

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards `?` and `*`.

For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

Why Cannot Silk Test Workbench Recognize Apache Flex Controls?

If Silk Test Workbench cannot recognize the controls of an Apache Flex application, which you are accessing through a Web server, you can try the following things:

- Compile your Apache Flex application with the Adobe automation libraries and the appropriate `FlexTechDomain.swc` for the Apache Flex version.
- Use runtime loading.
- Apache Flex controls are not recognized when embedding an Apache Flex application with an empty `id` attribute.

Java AWT/Swing Support

Silk Test Workbench provides built-in support for testing applications or applets that use the Java AWT/Swing controls. When you configure an application or applet that uses Java AWT/Swing, Silk Test Workbench automatically provides support for testing standard AWT/Swing controls.

Note: You can also test Java SWT controls embedded in Java AWT/Swing applications or applets as well as Java AWT/Swing controls embedded in Java SWT applications.

Note: Image click recording is not supported for applications or applets that use the Java AWT/Swing controls.

Sample Applications

Silk Test provides a sample Swing test application. Download and install the sample applications from *http://supportline.microfocus.com/websync/SilkTest.aspx*. After you have installed the sample applications, click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Sample Applications** > **Java Swing** > **Swing Test Application** or (in Microsoft Windows 10) **Start** > **Silk**.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Supported Controls

For a complete list of the controls available for Java AWT/Swing testing, see *Java Swing Class Reference*.

Attributes for Java AWT/Swing Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Java AWT/Swing include:

- caption
- priorlabel: Helps to identify text input fields by the text of its adjacent label field. Every input field of a form usually has a label that explains the purpose of the input. For controls that do not have a caption, the attribute **priorlabel** is automatically used in the locator. For the **priorlabel** value of a control, for example a text input field, the caption of the closest label at the left side or above the control is used.
- name
- accessibleName
- *Swing only*: All custom object definition attributes set in the widget with `putClientProperty("propertyName", "propertyValue")`

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Dynamically Invoking Java Methods

Dynamic invoke enables you to directly call methods, retrieve properties, or set properties, on an actual instance of a control in the application under test. You can also call methods and properties that are not available in the Silk Test Workbench API for this control. Dynamic invoke is especially useful when you are working with custom controls, where the required functionality for interacting with the control is not exposed through the Silk Test Workbench API.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

Call dynamic methods on objects with the `Invoke` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Call multiple dynamic methods on objects with the `InvokeMethods` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.

For example, to call a method named `SetTitle`, which requires the title to be set as an input parameter of type string, on an actual instance of a control in the application under test, type the following:

```
control.Invoke("SetTitle", "my new title")
```

Note: Typically, most properties are read-only and cannot be set.

Note: Reflection is used in most technology domains to call methods and retrieve properties.

Supported Methods and Properties

The following methods and properties can be called:

- Methods and properties that Silk Test Workbench supports for the control.
- All public methods of the SWT, AWT, or Swing widget
- If the control is a custom control that is derived from a standard control, all methods and properties from the standard control can be called.

Supported Parameter Types

The following parameter types are supported:

- Primitive types (boolean, integer, long, double, string)

  Both primitive types, such as `int`, and object types, such as `java.lang.Integer` are supported. Primitive types are widened if necessary, allowing, for example, to pass an `int` where a `long` is expected.
- Enum types

  Enum parameters must be passed as string. The string must match the name of an enum value. For example, if the method expects a parameter of the enum type, `java.sql.ClientInfoStatus` you can use the string values of `REASON_UNKNOWN`, `REASON_UNKNOWN_PROPERTY`, `REASON_VALUE_INVALID`, or `REASON_VALUE_TRUNCATED`
- Lists

Allows calling methods with list, array, or var-arg parameters. Conversion to an array type is done automatically, provided the elements of the list are assignable to the target array type.

- Other controls

  Control parameters can be passed or returned as TestObject.

Returned Values

The following values are returned for properties and methods that have a return value:

- The correct value for all built-in Silk Test Workbench types. These types are listed in the *Supported Parameter Types* section.
- All methods that have no return value return `null` in C# or `Nothing` in VB.

Determining the priorLabel in the Java AWT/Swing Technology Domain

To determine the priorLabel in the Java AWT/Swing technology domain, all labels and groups in the same window as the target control are considered. The decision is then made based upon the following criteria:

- Only labels either above or to the left of the control, and groups surrounding the control, are considered as candidates for a priorLabel.
- If a parent of the control is a `JViewPort` or a `ScrollPane`, the algorithm works as if the parent is the window that contains the control, and nothing outside is considered relevant.
- In the simplest case, the label closest to the control is used as the priorLabel.
- If two labels have the same distance to the control, and one is to the left and the other above the control, the left one is preferred.
- If no label is eligible, the caption of the closest group is used.

Oracle Forms Support

This functionality is supported only if you are using the Open Agent.

Silk Test Workbench provides built-in support for testing applications that are based on Oracle Forms.

Note: For some controls, Silk Test Workbench provides only low-level recording support.

For information on the supported versions and browsers for Oracle Forms, refer to the *Release Notes*.

Prerequisites for Testing Oracle Forms

To test an application that is built with Oracle Forms, the following prerequisites need to be fulfilled:

- The next-generation Java Plug-In needs to be enabled. This setting is enabled by default. You can change the setting in the **Java Control Panel**. For additional information on the next-generation Java Plug-In, refer to the Java documentation.
- To prevent Java security dialogs from displaying during a test run, the Applet needs to be signed.
- Micro Focus recommends enabling the `Names` property. When this property is enabled, the Oracle Forms runtime exposes the internal *name*, which is the name that the developer of the control has specified for the control, as the `Name` property of the control. Otherwise, the `Name` property will hold a calculated value, which usually consists of the class name of the control plus an index. This enables Silk Test Workbench to generate stable locators for controls.

Attributes for Oracle Forms Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Oracle Forms include:

- priorlabel: Helps to identify text input fields by the text of its adjacent label field. Every input field of a form usually has a label that explains the purpose of the input. For controls that do not have a caption, the attribute **priorlabel** is automatically used in the locator. For the **priorlabel** value of a control, for example a text input field, the caption of the closest label at the left side or above the control is used.
- name
- accessibleName

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Java SWT and Eclipse RCP Support

Silk Test provides built-in support for testing applications that use widgets from the Standard Widget Toolkit (SWT) controls. When you configure a Java SWT/RCP application, Silk Test automatically provides support for testing standard Java SWT/RCP controls.

Silk Test supports:

- Testing Java SWT controls embedded in Java AWT/Swing applications as well as Java AWT/Swing controls embedded in Java SWT applications.
- Testing Java SWT applications.
- Any Eclipse-based application that uses SWT widgets for rendering. Silk Test supports both Eclipse IDE-based applications and RCP-based applications.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Supported Controls

For a complete list of the widgets available for SWT testing, see *Java SWT Class Reference*.

Java SWT Class Reference

When you configure a Java SWT application, Silk Test Workbench automatically provides built-in support for testing standard Java SWT controls.

Java SWT Custom Attributes

You can add custom attributes to a test application to make a test more stable. For example, in Java SWT, the developer implementing the GUI can define an attribute (for example, `'silkTestAutomationId'`) for a widget that uniquely identifies the widget in the application. A tester using Silk Test Workbench can then add that attribute to the list of custom attributes (in this case, `'silkTestAutomationId'`), and can identify controls by that unique ID. Using a custom attribute is more reliable than other attributes like caption or index, since a caption will change when you translate the application into another language, and the index will change whenever another widget is added before the one you have defined already.

If more than one object is assigned the same custom attribute value, all the objects with that value will return when you call the custom attribute. For example, if you assign the unique ID, `'loginName'` to two different text fields, both fields will return when you call the `'loginName'` attribute.

Java SWT Example

If you create a button in the application that you want to test using the following code:

```
Button myButton = Button(parent, SWT.NONE);

myButton.setData("SilkTestAutomationId", "myButtonId");
```

To add the attribute to your XPath query string in your test, you can use the following query:

```
Dim button =
desktop.PushButton("@SilkTestAutomationId='myButton'")
```

To enable a Java SWT application for testing custom attributes, the developers must include custom attributes in the application. Include the attributes using the `org.swt.widgets.Widget.setData(String key, Object value)` method.

Attributes for Java SWT Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Java SWT include:

- caption
- all custom object definition attributes

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Dynamically Invoking Java Methods

Dynamic invoke enables you to directly call methods, retrieve properties, or set properties, on an actual instance of a control in the application under test. You can also call methods and properties that are not available in the Silk Test Workbench API for this control. Dynamic invoke is especially useful when you are working with custom controls, where the required functionality for interacting with the control is not exposed through the Silk Test Workbench API.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

Call dynamic methods on objects with the `Invoke` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Call multiple dynamic methods on objects with the `InvokeMethods` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.

For example, to call a method named `SetTitle`, which requires the title to be set as an input parameter of type string, on an actual instance of a control in the application under test, type the following:

```
control.Invoke("SetTitle", "my new title")
```

Note: Typically, most properties are read-only and cannot be set.

Note: Reflection is used in most technology domains to call methods and retrieve properties.

Supported Methods and Properties

The following methods and properties can be called:

- Methods and properties that Silk Test Workbench supports for the control.
- All public methods of the SWT, AWT, or Swing widget
- If the control is a custom control that is derived from a standard control, all methods and properties from the standard control can be called.

Supported Parameter Types

The following parameter types are supported:

- Primitive types (boolean, integer, long, double, string)

  Both primitive types, such as `int`, and object types, such as `java.lang.Integer` are supported. Primitive types are widened if necessary, allowing, for example, to pass an `int` where a `long` is expected.
- Enum types

  Enum parameters must be passed as string. The string must match the name of an enum value. For example, if the method expects a parameter of the enum type, `java.sql.ClientInfoStatus` you can use the string values of `REASON_UNKNOWN`, `REASON_UNKNOWN_PROPERTY`, `REASON_VALUE_INVALID`, or `REASON_VALUE_TRUNCATED`
- Lists

  Allows calling methods with list, array, or var-arg parameters. Conversion to an array type is done automatically, provided the elements of the list are assignable to the target array type.
- Other controls

  Control parameters can be passed or returned as TestObject.

Returned Values

The following values are returned for properties and methods that have a return value:

- The correct value for all built-in Silk Test Workbench types. These types are listed in the *Supported Parameter Types* section.
- All methods that have no return value return `null` in C# or `Nothing` in VB.

Troubleshooting Java SWT and Eclipse Applications

Some SWTTree methods do not replay with low-level playback

When using low-level playback, some SWTTree methods, for example `Expand` and `Collapse`, do not replay.

To solve this problem, set the replay mode to **Default**. For additional information, see *Setting General Playback Options*.

Selecting a non-visible node in an SWTTree

When using low-level playback, Silk Test Workbench cannot interact with non-visible nodes in an SWTTree.

To solve this problem, set the replay mode to **Default**. For additional information, see *Setting General Playback Options*.

Testing Mobile Applications

Silk Test Workbench enables you to automatically test your native mobile applications (apps) and mobile web applications. Automatically testing your mobile applications with Silk Test Workbench provides the following benefits:

- It can significantly reduce the testing time of your mobile applications.
- You can create your tests once and then test your mobile applications on a large number of different devices and platforms.
- You can ensure the reliability and performance that is required for enterprise mobile applications.
- It can increase the efficiency of QA team members and mobile application developers.
- Manual testing might not be efficient enough for an agile-focused development environment, given the large number of mobile devices and platforms on which a mobile application needs to function.

Note: To test native mobile applications or hybrid applications with Silk Test Workbench, you require a native mobile license. For additional information, see *Licensing Information*.

Note: Silk Test Workbench provides support for testing mobile apps on both Android and iOS devices.

For information on the supported operating system versions and the supported browsers for testing mobile applications, refer to the *Release Notes*.

Android

Silk Test Workbench enables you to test a mobile application on an Android device or an Android emulator.

Prerequisites for Testing Mobile Applications on Android

Before you can test a mobile application (app) on an Android device or on an Android emulator, ensure that the following prerequisites are met:

- Ensure that Java is installed on the machine on which Silk Test Workbench is running, and that the path to Java is added to the *Path* environment variable. Silk Test Workbench requires either a JRE or a JDK for mobile testing. If Java is not installed, add `C:\Program Files (x86)\Silk\SilkTest\ng\jre\bin` to the *Path*.
- If you have created your own hybrid app by adding a web view to a native mobile app, add the following code to the app to make your app testable with Silk Test Workbench:

```
WebView.setWebContentsDebuggingEnabled(true);
webView.getSettings().setJavaScriptEnabled(true);
```

- Silk Test Workbench does not support showing a live view of the device screen for Android 4.4. Micro Focus recommends using Android 5 or later.

Testing Mobile Applications on Android

To test a mobile application on a physical Android device or on an Android emulator, perform the following tasks:

1. Ensure that you have met the prerequisites for testing mobile applications on Android.

   For additional information, see *Prerequisites for Testing Mobile Applications on Android*.
2. If you want to test the mobile application on an Android emulator, configure the emulator settings for Silk Test Workbench.

   For additional information, see *Configuring the Android Emulator for Silk Test*.
3. Start the Android emulator or connect the device to the machine on which Silk Test Workbench is installed.

4. If you want to test the mobile application on a physical Android device that you are using for the first time on this machine, install the appropriate Android USB Driver on the machine.
   For additional information, see *Installing a USB Driver*.
5. If you want to test the mobile application on a physical Android device, enable USB-debugging on the Android device.
   For additional information, see *Enabling USB-Debugging*.
6. Create a Silk Test Workbench project for your mobile application.
7. Create a test for your mobile application.
8. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
9. To test a mobile web application:
   a) Select the **Web** tab.
   b) Select the mobile browser that you want to use.
   c) Specify the web page to open in the **Enter URL to navigate** text box.
10. To test a native mobile application or a Hybrid application:

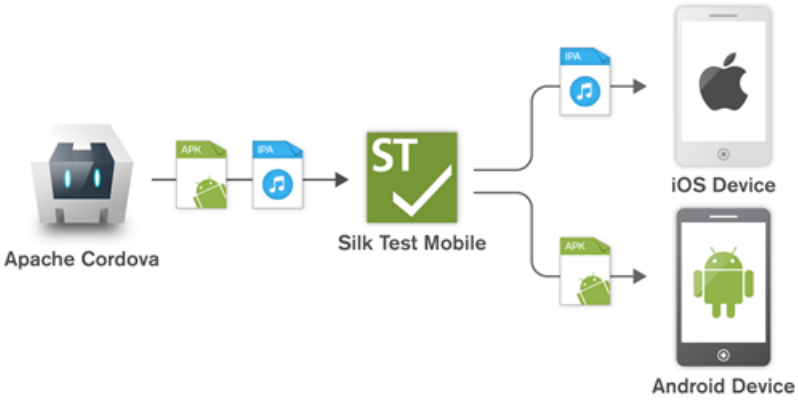
    Note: To test native mobile applications or hybrid applications with Silk Test Workbench, you require a native mobile license. For additional information, see *Licensing Information*.

    a) Select the **Mobile** tab.
    b) Select the mobile device, on which you want to test the app, from the list.
    c) Click **Browse** to select the app file or enter the full path to the app file into the **Mobile app file** text field.
       Silk Test Workbench supports HTTP and UNC formats for the path.
       Silk Test Workbench installs the app on the mobile device or emulator.
11. Click **OK**.
    An Android device or emulator must not be screen-locked during testing. To keep the device awake while it is connected to a machine, open the settings and tap **Developer Options**. Then check **Stay awake** or **Stay awake while charging**.
12. Use the **Recording** window to record the test against the mobile application.
    For additional information, see *Recording Mobile Applications*.
13. When you have recorded all the actions, stop the recording.
14. Replay the test.
15. Analyze the test results.

Testing Hybrid Applications on Android

Hybrid applications (apps) are apps that are run on the device, like native applications, but are written with web technologies, for example HTML5, CSS, and JavaScript.

Silk Test Workbench provides full browser support for testing debug hybrid apps that consist of a single web view, which is embedded in a native container. A common example of such a hybrid app would be an Apache Cordova application.



To prepare a non-debug hybrid app for testing, enable remote debugging in the app by adding the following code to the app:

```
WebView.setWebContentsDebuggingEnabled(true);
webView.getSettings().setJavaScriptEnabled(true);
```

To test non-debug hybrid apps without remote debugging enabled or hybrid apps that include more than one web view, enable the Silk Test Workbench fallback support by setting the option OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT to TRUE. For additional information, see *Setting Advanced Options*. With the fallback support enabled, Silk Test Workbench recognizes and handles the controls in a web view as native mobile controls instead of browser controls. For example, the following code clicks on a link when using browser support:

```
' VB .NET code
Agent.SetOption(Options.EnableMobileWebviewFallbackSupport, False)
_desktop.DomLink("//BrowserApplication//BrowserWindow//
INPUT[@id='email']").Click()
```

With the fallback support enabled, the following code clicks on the same link:

```
' VB .NET code
Agent.SetOption(Options.EnableMobileWebviewFallbackSupport, True)
_desktop.Find("//BrowserApplication//BrowserWindow//MobileTextField[@resource-
id='email']").Click()
```

Silk Test Workbench can detect web views that support Chrome remote debugging. Silk Test Workbench can detect web views with either the package *com.android.webview* or the package *com.google.android.webview*, which are the default packages on most Android devices.

Note: Silk Test Workbench supports testing hybrid apps on Android 4.4 or later. To test hybrid apps on Android, Android System WebView version 51 or later is required.

The process for testing a hybrid app on Android is the same as the process for testing a mobile native application. For additional information, see *Testing Mobile Applications on Android*.

Installing a USB Driver

To connect an Android device for the first time to your local machine to test your mobile applications, you need to install the appropriate USB driver.

The device manufacturer might provide an executable with all the necessary drivers for the device. In this case you can just install the executable on your local machine. If the manufacturer does not provide such an executable, you can install a single USB driver for the device on the machine.

To install the Android USB driver:

1. Download the appropriate driver for your device.

   For example, for information on finding and installing a USB driver for a Google Nexus device, see *http://developer.android.com/tools/extras/oem-usb.html*.
2. Connect your Android device to a USB port on your local machine.
3. From your desktop or **Windows Explorer**, right-click **Computer** and select **Manage**.
4. In the left pane, select **Device Manager**.
5. In the right pane, locate and expand **Other device**.
6. Right-click the device name, for example *Nexus 5x*, and select **Update Driver Software**. The **Hardware Update Wizard** opens.
7. Select **Browse my computer for driver software** and click **Next**.
8. Click **Browse** and navigate to the folder to which you have downloaded the USB driver.
9. Select the USB driver.
10. Click **Next** to install the driver.

Enabling USB-Debugging

To communicate with an Android device over the Android Debug Bridge (adb), enable USB debugging on the device.

1. On the Android device, open the settings.
2. Tap **Developer Settings**.

   The developer settings are hidden by default. If the developer settings are not included in the settings menu of the device:

   a) Depending on whether the device is a phone or a pad, scroll down and tap **About phone** or **About Pad**.

   b) Scroll down again and tap **Build Number** seven times.
3. In the **Developer settings** window, check **USB-Debugging**.
4. Set the USB mode of the device to **Media device (MTP)**, which is the default setting.

   For additional information, refer to the documentation of the device.

Recommended Settings for Android Devices

To optimize testing with Silk Test Workbench, configure the following settings on the Android device that you want to test:

- Enable USB-debugging on the Android device. For additional information, see *Enabling USB-Debugging*
- An Android device must be connected as a media device to the machine on which the Open Agent is running. The USB mode of the Android device must be set to **Media device (MTP)**.
- An Android device or emulator must not be screen-locked during testing. To keep the device awake while it is connected to a machine, open the settings and tap **Developer Options**. Then check **Stay awake** or **Stay awake while charging**.

Configuring the Android Emulator for Silk Test Workbench

Note: When using an Android emulator, an additional adb server is running in addition to the one that is used by Silk Test Workbench. If the running adb servers have different versions, the connection between the Open Agent and the device might become unstable or even break. To avoid version mismatch errors, specify the path to the Android SDK directory by setting the environment variable *SILK_ANDROID_HOME*, for example to `C:\Users\<user>\AppData\Local\Android \android-sdk`. If the information service was running during this change, use the Windows Service Manager to restart the Silk Test information service with the updated environment variable. If the environment variable is not set, Silk Test Workbench uses the adb version that is shipped with Silk Test Workbench.

When you want to test mobile applications on an Android emulator with Silk Test Workbench, you have to configure the emulator for testing:

1. Install the latest version of the Android SDK.

   For information on how to install and configure the Android SDK, see *Get the Android SDK*.
2. Install Android Studio 2.

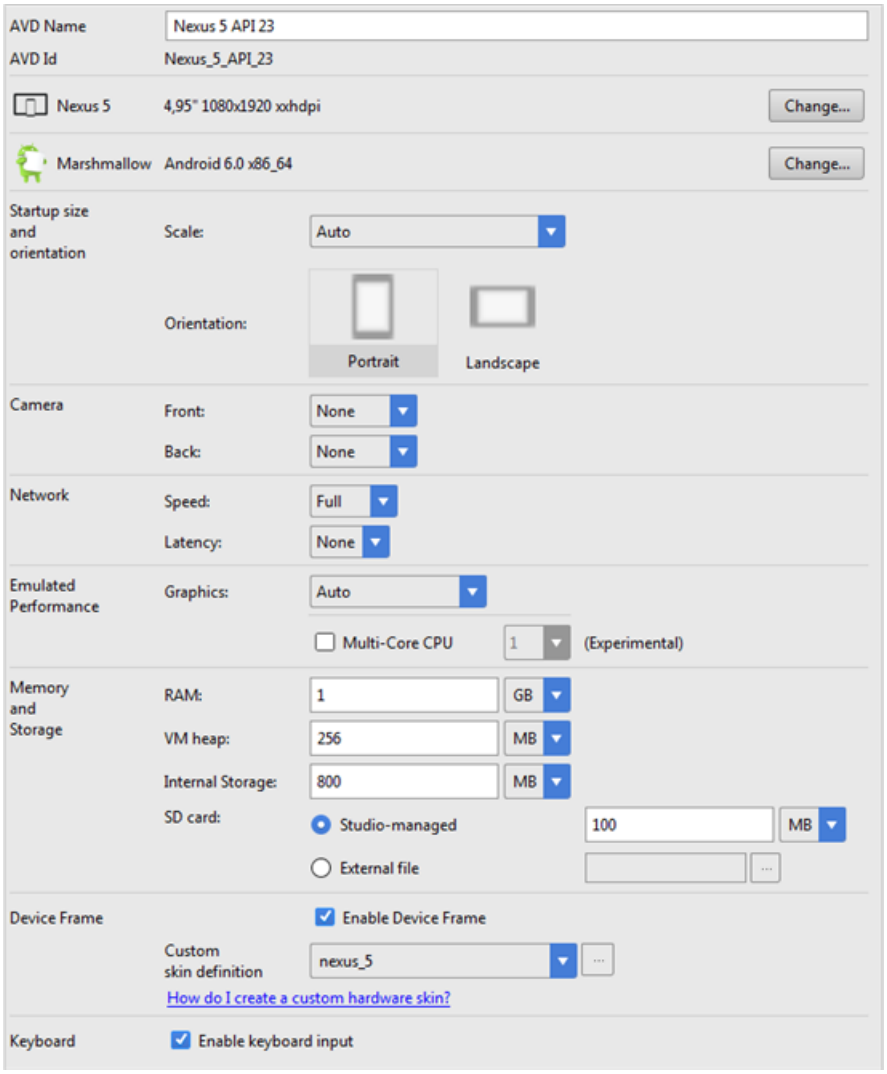
   

   Tip: You can skip installing Android Studio 2 and use the emulator provided with the Android SDK. However, Micro Focus recommends installing Android Studio 2 for improved emulator performance. The remaining steps in this topic require Android Studio 2 to be installed.
3. From Android Studio 2, start the **AVD Manager**.
4. Click **Create Virtual Device**.
5. Select a virtual device.
6. Click **Next**.

7. Download and select a system image of Android that includes Google APIs.
8. Click **Next**.
9. Configure the virtual device according to your requirements.
10. Click **Show Advanced Settings**.
11. Adjust the RAM size and the heap space used by the emulator to an amount that is manageable by your machine.

    Tip: Micro Focus recommends using at least 1 GB RAM and 256 MB heap space.

12. Select **Auto** from the list in the **Emulated Performance** area.

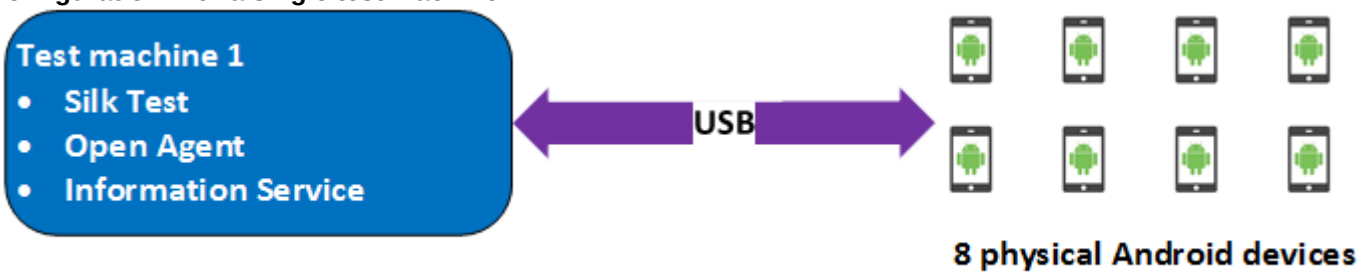


13. Click **Finish**.

Tested Configurations for Parallel Test Execution

With Silk Test Workbench, you can run automated tests on multiple Android devices in parallel. The amount of Android devices that you are able to use in parallel depends on the available hardware. Micro Focus has successfully tested the following hardware configurations:

Configuration with a single test machine

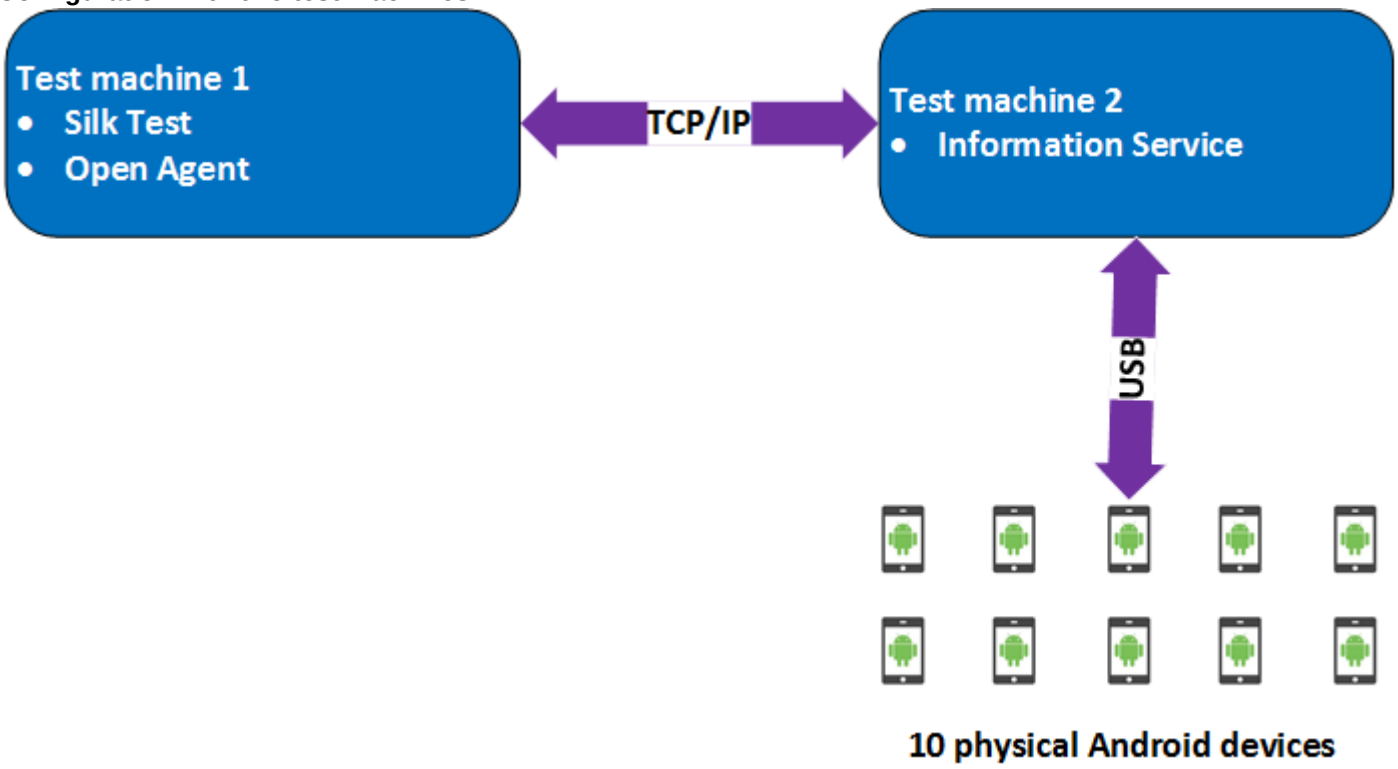


Using a single test machine directly connected to the Android devices through USB, we tested up to 8 physical Android devices in parallel.

The test machine was a Lenovo ThinkPad T450 with the following hardware specifications:

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 cores (4 threads)
- 8 GB RAM

Configuration with two test machines



Here we are using two test machines, one with Silk Test Workbench installed and another, which is configured as a remote location for the first machine and has the Silk Test Information Service installed. Using such a configuration, we tested up to 10 physical Android devices in parallel.

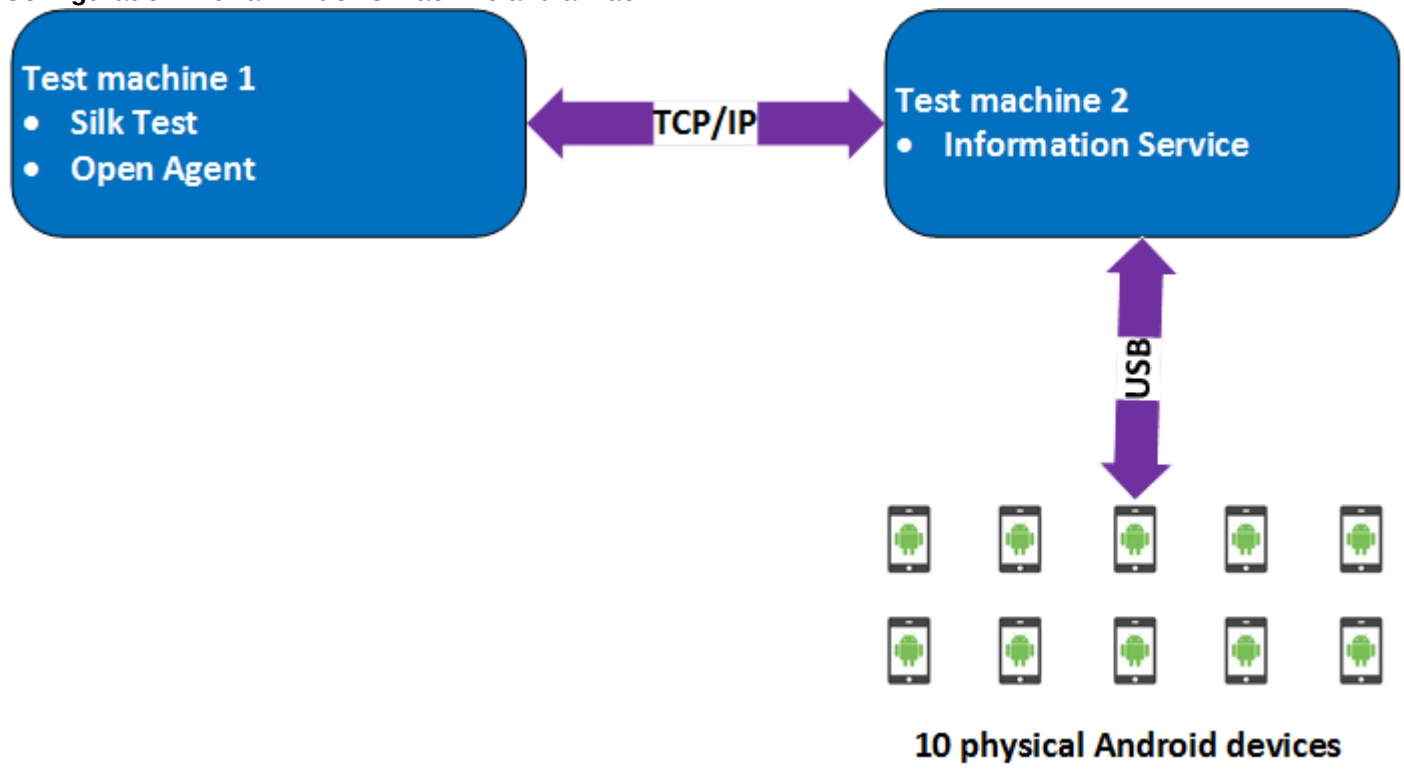
Test machine 1 was a Lenovo ThinkPad T450 with the following hardware specifications:

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 cores (4 threads)
- 8 GB RAM

Test machine 2 was a Dell Precision T1700 with the following hardware specifications:

- Intel® Core™ i7 - 4770 CPU @ 3.40 GHz
- 4 cores (8 threads)
- 16 GB RAM

Configuration with a Windows machine and a Mac



Here we are using two test machines, a Windows machine with Silk Test Workbench installed and a Mac, which is configured as a remote location for the first machine and has the Silk Test Information Service installed. Using such a configuration, we tested up to 10 physical Android devices in parallel.

Test machine 1 was a Lenovo ThinkPad T450 with the following hardware specifications:

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 cores (4 threads)
- 8 GB RAM

Test machine 2 was an Apple Mac Mini with the following hardware specifications:

- Intel® Core™ i5 - 4782U CPU @ 2.60 GHz
- 2 cores (4 threads)
- 16 GB RAM

iOS

Silk Test Workbench enables you to test a mobile application on an iOS device or an iOS Simulator.

Because of significant changes by Apple in iOS 9.3 in comparison to the previous versions of iOS, Silk Test supports testing mobile applications on iOS 9.3 or later. For a list of the supported iOS versions, refer to the *Release Notes*.

Note: Testing mobile applications on iOS 11 requires Xcode 9. When using Xcode 9 on a Mac, testing on physical devices and Simulators with iOS versions prior to iOS 11 that are connected to or running

on this Mac is not supported. Use Xcode 8.3 to test physical devices and Simulators with iOS 9.3 and iOS 10.

Tip: To test on iOS versions prior to iOS 9.3, you can use Silk Test 17.5. The following table shows the major changes when testing on iOS with Silk Test 19.0 in comparison to Silk Test 17.5:

Silk Test 19.0	Silk Test 17.5
Supports iOS 9.3, iOS 10, and iOS 11.	Supports iOS 8.1, 8.2, 8.3, 8.4, 9.0, 9.1, 9.2, and 9.3
Supports Xcode 8.3 and Xcode 9.	Supports Xcode 6 and Xcode 7.
Supports testing multiple physical iOS devices in the same user session. **Tip:** If you have created multiple user sessions to test with Silk Test 17.5, Micro Focus recommends removing all user sessions except one.	Requires multiple user sessions on a Mac to test multiple physical iOS devices.

Prerequisites for Testing Mobile Applications on iOS

Before you can test a mobile application (app) on an iOS device or on an iOS Simulator, ensure that the following prerequisites are met:

- The current version of the Silk Test information service is installed on the Mac. For additional information, see *Installing the Silk Test Infoservice on a Mac*.
- If you want to test your application on a physical iOS device, ensure the following:
  - The device is connected to the Mac.
  - The device has a supported version of iOS. For a list of the supported iOS versions, refer to the *Release Notes*.
- If you want to test your application on an iOS Simulator, ensure the following:
  - The iOS Simulator image is installed on the Mac.
  - The iOS Simulator image has a supported version of iOS. For a list of the supported iOS versions, refer to the *Release Notes*.
- If you want to test your application on an physical iOS device, ensure that the same time zone is set on the device and the Mac.
- A supported version of Xcode is installed on the Mac.
- Silk Test Workbench is installed on a Windows machine.
- The Mac is located in the same network as the Windows machine and is added as a remote location to the Windows machine.
- To test a native mobile app on an iOS device, ensure that the `.ipa` file of your app has been signed with a developer account. For additional information, see *Preparing an iOS App for Testing*.
- To test a native mobile app on an iOS Simulator, ensure that the app has been zipped. For additional information, see *Testing Native Mobile Applications on an iOS Simulator*.
- To test a native mobile app on both an iOS device and an iOS Simulator, ensure that both the signed `.ipa` file and the zipped `.app` directory are located in the same folder.
- To test a native mobile app, ensure that the ID of the iOS device is associated with the developer profile which was used to sign the app.
- The iOS device must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings** > **General** > **Passcode Lock**.
- The Mac should not switch off the screen during testing, otherwise the **Playback Status** dialog box will not display anything.
- To test a mobile application on an iOS Simulator, deactivate the display sleep on the Mac during testing.
- To test a native mobile app on a physical iOS device, enable the UI automation on the device. For additional information, see *Preparing an iOS Device for Testing*.
- To test a mobile web application with Apple Safari on a physical iOS device, activate the **Web Inspector**. For additional information, see *Preparing an iOS Device for Testing*.

- Micro Focus recommends using iOS devices which have a Lightning connector. Silk Test Workbench does not support showing a live view of the device screen for iOS devices that are not connected to a Mac through a Lightning cable.

Testing Native Mobile Applications on a Physical iOS Device

Note: To test native mobile applications or hybrid applications with Silk Test Workbench, you require a native mobile license. For additional information, see *Licensing Information*.

For information on the prerequisites for testing mobile applications on iOS, see *Prerequisites for Testing Mobile Applications on iOS*. For information on the known limitations when testing native mobile applications, see *Limitations for Testing Mobile Native Applications*.

To test a native mobile application (app) or a hybrid application on a physical iOS device, perform the following tasks:

1. Prepare the iOS device for testing.

   For additional information, see *Preparing an iOS Device for Testing*.
2. Prepare the app for testing.

   For additional information, see *Preparing an iOS App for Testing*.
3. Prepare the Mac for testing. For additional information, see *Preparing a Mac for Testing Mobile Applications on iOS*.
4. Add the Mac, to which the iOS device is connected, as a remote location to the Windows machine on which Silk Test is installed.

   For additional information, see *Editing Remote Locations*.

   Note: At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac. With Silk Test 17.5 Hotfix 1 or later, you are no longer required to use multiple user sessions on a Mac to test mobile applications on iOS.
5. Create a Silk Test Workbench project for your mobile application.
6. Create a test for your mobile application.
7. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
8. Select the **Mobile** tab.
9. Select the mobile device, on which you want to test the app, from the list.
10. Click **Browse** to select the app file or enter the full path to the app file into the **Mobile app file** text field.

    Silk Test Workbench supports HTTP and UNC formats for the path.

    Silk Test Workbench installs the app on the mobile device.
11. Click **OK**.

    An iOS device or Simulator must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings** > **General** > **Passcode Lock**.
12. When you have recorded all actions, stop recording.
13. Replay the test.
14. Analyze the test results.

Note: To test a native mobile app on both an iOS device and an iOS Simulator, ensure that both the signed `.ipa` file and the zipped `.app` directory are located in the same folder.

Testing Native Mobile Applications on an iOS Simulator

Note: To test native mobile applications or hybrid applications with Silk Test Workbench, you require a native mobile license. For additional information, see *Licensing Information*.

For information on the prerequisites for testing mobile applications on iOS, see *Prerequisites for Testing Mobile Applications on iOS*. For information on the known limitations when testing native mobile applications, see *Limitations for Testing Mobile Native Applications*.

To test a native mobile application (app) or a hybrid application on an iOS Simulator, perform the following tasks:

1. Prepare the Mac for testing. For additional information, see *Preparing a Mac for Testing Mobile Applications on iOS*.
2. In the Xcode project of the app, compile the app for the iOS Simulator.

   You can compile the app either from the Xcode UI or from the command line. For example, to compile the app through the command line for an iOS Simulator with iOS 10.0, execute the following command:

   ```
   xcodebuild -sdk iphonesimulator10.0
   ```
3. Zip up the `.app` directory of the app into a `.zip` file.

   By default, the .app directory is located in the directory `~/Library/Developer/Xcode/DerivedData`. You can click **File** > **Project Settings** in Xcode to see into which location the directory is stored.
4. Add the Mac, on which the iOS Simulator is installed, as a remote location to the Windows machine on which Silk Test Workbench is installed.

   For additional information, see *Editing Remote Locations*.

   Note: You can only test on one iOS Simulator that is installed on a Mac. Multiple Silk Test Workbench users cannot simultaneously test on multiple iOS Simulators that are installed on the same Mac.
5. Create a Silk Test Workbench project for your mobile application.
6. Create a test for your mobile application.
7. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
8. Select the **Mobile** tab.
9. Select the iOS Simulator from the list.
10. Click **Browse** to select the zipped app file or enter the full path to the zipped app file into the **Mobile app file** text field.

    Silk Test Workbench supports HTTP and UNC formats for the path.

    Silk Test Workbench installs the app on the iOS Simulator.
11. Click **OK**.

    An iOS device or Simulator must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings** > **General** > **Passcode Lock**.
12. When you have recorded all actions, stop recording.
13. Replay the test.
14. Analyze the test results.

Note: To test a native mobile app on both an iOS device and an iOS Simulator, ensure that both the signed `.ipa` file and the zipped `.app` directory are located in the same folder.

Testing Mobile Web Applications on a Physical iOS Device

For information on the prerequisites for testing mobile applications on iOS, see *Prerequisites for Testing Mobile Applications on iOS*. For information on the known limitations when testing mobile web applications, see *Limitations for Testing Mobile Web Applications*.

To test a mobile web application on a physical iOS device, perform the following tasks:

1. Prepare the iOS device for testing.

   For additional information, see *Preparing an iOS Device for Testing*.

2. Prepare the Mac for testing. For additional information, see *Preparing a Mac for Testing Mobile Applications on iOS*.
3. Add the Mac, to which the iOS device is connected, as a remote location to the Windows machine on which Silk Test is installed.

   For additional information, see *Editing Remote Locations*.

   Note: At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac. With Silk Test 17.5 Hotfix 1 or later, you are no longer required to use multiple user sessions on a Mac to test mobile applications on iOS.
4. Create a Silk Test Workbench project for your mobile application.
5. Create a test for your mobile application.
6. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
7. To test a mobile web application:
   a) Select the **Web** tab.
   b) Select the mobile browser that you want to use.
   c) Specify the web page to open in the **Enter URL to navigate** text box.
8. Click **OK**.

   An iOS device or Simulator must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings** > **General** > **Passcode Lock**.
9. When you have recorded all actions, stop recording.
10. Replay the test.
11. Analyze the test results.

Testing Mobile Web Applications on an iOS Simulator

For information on the known limitations when testing mobile web applications, see *Limitations for Testing Mobile Web Applications*.

To test a mobile web application on an iOS Simulator, perform the following tasks:

1. Prepare the Mac for testing. For additional information, see *Preparing a Mac for Testing Mobile Applications on iOS*.
2. Add the Mac, on which the iOS Simulator is installed, as a remote location to the Windows machine on which Silk Test is installed.

   For additional information, see *Editing Remote Locations*.

   Note: At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac. With Silk Test 17.5 Hotfix 1 or later, you are no longer required to use multiple user sessions on a Mac to test mobile applications on iOS.
3. Create a Silk Test Workbench project for your mobile application.
4. Create a test for your mobile application.
5. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
6. To test a mobile web application:
   a) Select the **Web** tab.
   b) Select the mobile browser that you want to use.
   c) Specify the web page to open in the **Enter URL to navigate** text box.
7. Click **OK**.

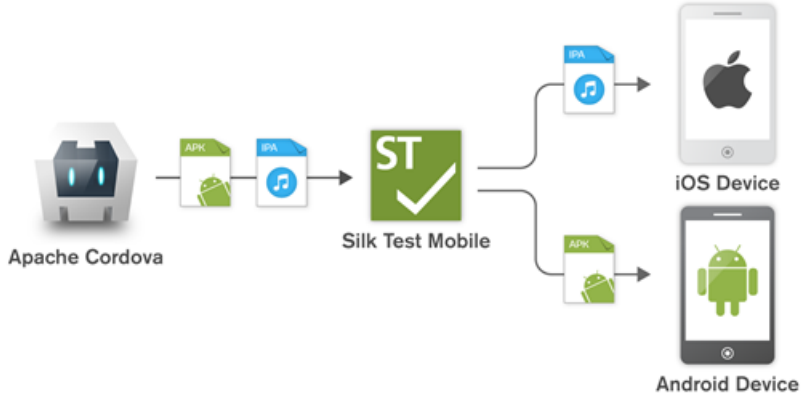
   An iOS device or Simulator must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings** > **General** > **Passcode Lock**.

8. When you have recorded all actions, stop recording.
9. Replay the test.
10. Analyze the test results.

Testing Hybrid Applications on iOS

Hybrid applications (apps) are apps that are run on the device, like native applications, but are written with web technologies, for example HTML5, CSS, and JavaScript.

Silk Test Workbench provides full browser support for testing debug hybrid apps that consist of a single web view, which is embedded in a native container. A common example of such a hybrid app would be an Apache Cordova application.



To test non-debug hybrid apps without remote debugging enabled or hybrid apps that include more than one web view, enable the Silk Test Workbench fallback support by setting the option OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT to TRUE. For additional information, see *Setting Advanced Options*. With the fallback support enabled, Silk Test Workbench recognizes and handles the controls in a web view as native mobile controls instead of browser controls. For example, the following code clicks on a link when using browser support:

```
' VB .NET code
Agent.SetOption(Options.EnableMobileWebviewFallbackSupport, False)
_desktop.DomLink("//BrowserApplication//BrowserWindow//
INPUT[@id='email']").Click()
```

With the fallback support enabled, the following code clicks on the same link:

```
' VB .NET code
Agent.SetOption(Options.EnableMobileWebviewFallbackSupport, True)
_desktop.Find("//BrowserApplication//BrowserWindow//MobileTextField[@resource-
id='email']").Click()
```

The process for testing a hybrid app on iOS is the same as the process for testing a mobile native application. For additional information, see *Testing Native Mobile Applications on a Physical iOS Device* or *Testing Native Mobile Applications on an iOS Simulator*.

Before testing a hybrid app on an iOS device, ensure that the **Web Inspector** is activated on the device.

Preparing an iOS Device for Testing

Note: To test native mobile applications or hybrid applications with Silk Test Workbench, you require a native mobile license. For additional information, see *Licensing Information*.

To prepare the iOS device to test mobile applications:

1. Start Xcode on the Mac.
2. Connect the iOS device to the Mac.

3. On the iOS device, click **Settings** > **Developer**.

   Tip: If the **Developer** menu is not displayed on the iOS device, restart the device and the Mac.

4. Activate **Enable UI Automation**.
5. To test a mobile web application on Apple Safari, click **Settings** > **Safari** > **Advanced**.
6. Activate the **Web Inspector**.
7. If you want to test on an iOS Simulator, enable **Rotate Device Automatically**.

   You can enable this setting by using the **Silk Test Configuration Assistant** or by enabling it manually. To open the **Configuration Assistant** on a Mac, click on the Silk Test icon in the status menus and select **Configuration Assistant**. To enable the setting manually, perform the following actions:

   a) On the Mac, start the iOS Simulator.
   b) With Xcode 9 or later, expand the **Hardware** menu.

      With prior versions of Xcode, expand the **Debug** menu.
   c) Check **Rotate Device Automatically**.
8. If you want to test on an iOS Simulator with Xcode 9 or later, disable **Show Device Bezels**.

   You can disable this setting by using the **Silk Test Configuration Assistant** or by disabling it manually. To open the **Configuration Assistant** on a Mac, click on the Silk Test icon in the status menus and select **Configuration Assistant**. To disable the setting manually, perform the following actions:

   a) On the Mac, start the iOS Simulator.
   b) Expand the **Window** menu.
   c) Uncheck **Show Device Bezels**.

Preparing an iOS App for Testing

To be able to test a specific iOS app on a specific iOS device with Silk Test Workbench, consider the following:

- Test automation is only possible with iOS apps that can be installed manually on specific iOS devices. To be able to sign an iOS app, you require a membership in the *Apple Developer Program*. For additional information, see *Choosing a Membership*. To test without having a membership in the *Apple Developer Program*, see *Using a Personal Team Profile for Testing on Physical iOS Devices*.

  Note: You cannot automatically test iOS apps that are created for publication in the App Store, as well as apps that can be installed manually on any iOS device.
- Before you can install and execute an iOS app on a specific iOS device, you have to register the iOS device with your Apple Developer account.
- You have to use Xcode to create an IPA file of the iOS app, which you can then install on the iOS device. To create IPA files for testing on a specific iOS device, members of the Apple Developer Program can use the *Archive* mechanism of Xcode, by using one of the following two options:

  - If you are a member of the *Apple Developer Enterprise Program*, you can use the **Save for Ad Hoc Deployment** option.
  - If you are a member of the Apple Developer Program, but not of the Apple Developer Enterprise Program, you can use the **Save for Development Deployment** option.

  For additional information, see *Exporting Your App for Testing (iOS, tvOS, watchOS)*.

To be able to test a specific iOS app on an iOS Simulator with Silk Test Workbench, use Xcode to create a ZIP file of the iOS app, which you can then install on the iOS Simulator. For additional information, refer to the Xcode documentation.

Installing the Silk Test Information Service on a Mac

Note: To install the information service on a Mac, you require administrative privileges on the Mac.

To create and execute tests against Apple Safari on a Mac, or against mobile applications on an iOS or Android device that is connected to a Mac, install the Silk Test information service (information service) on the Mac, and then use the **Remote Locations** dialog box to connect a Windows machine, on which Silk Test Workbench is installed, to the Mac.

To install the information service on a Mac:

1. Ensure that a Java JDK is installed on the Mac.
2. If you want to test mobile applications on an iOS device, ensure that Xcode is installed on the Mac.
3. Access the information service setup file, `SilkTestInformationService<Version>-<Build Number>.pkg`.
   - If you have downloaded the information service setup file while installing Silk Test, open the folder `macOS` in the Silk Test installation directory, for example `C:\Program Files (x86)\Silk\SilkTest`.
   - If you have not downloaded the information service setup file while installing Silk Test, you can download the setup file from *Micro Focus SupportLine*.
4. Copy the file `SilkTestInformationService<Version>-<Build Number>.pkg` to the Mac.
5. Execute `SilkTestInformationService<Version>-<Build Number>.pkg` to install the information service.
6. Follow the instructions in the installation wizard.
7. When asked for the password, provide the password of the currently signed in Mac user.
8. When Apple Safari opens and a message box asks whether to trust the SafariDriver, click **Trust**.

Note: You can only install the SafariDriver if you are directly logged in to the Mac, and not connected through a remote connection.

To complete the installation, the installer logs the current Mac user out. To verify that the information service was installed correctly, log in to the Mac and click on the Silk Test icon in the top-right corner of the screen to see the available devices and browsers.

Tip: If the Silk Test icon does not appear, restart the Mac.

Preparing a Mac to Test Mobile Applications on iOS

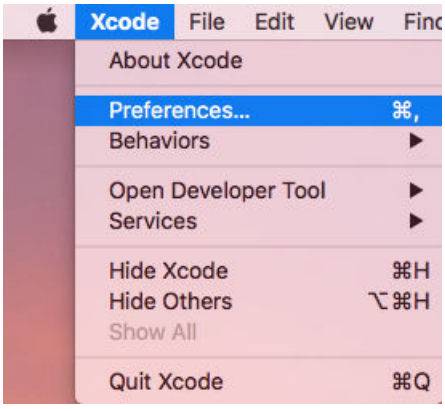
Note: To test native mobile applications or hybrid applications with Silk Test Workbench, you require a native mobile license. For additional information, see *Licensing Information*.

To test mobile applications on iOS, you require a Mac to which you can connect the iOS device, or on which the iOS Simulator is running. This Mac requires Xcode to be installed. For additional information on the prerequisites for testing mobile applications on iOS, see *Prerequisites for Testing Mobile Applications on iOS*.

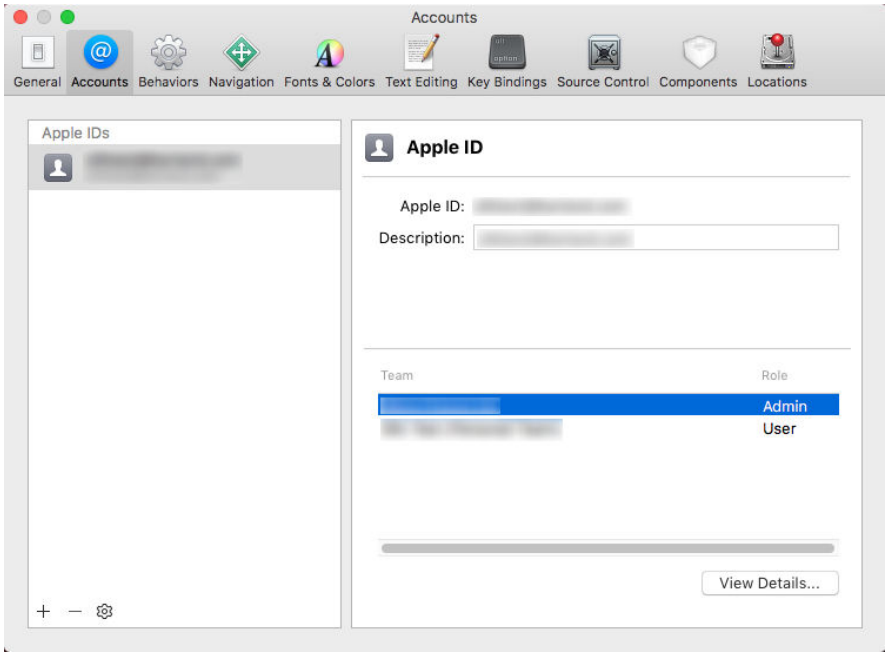
To execute iOS tests on a physical iOS device, follow the instructions in the **Silk Test Configuration Assistant** to configure the `WebDriverAgentRunner` Xcode project. To open the **Configuration Assistant**, click on the Silk Test icon in the status menus and select **Configuration Assistant**.

If for any reason you want to manually build the `WebDriverAgentRunner` Xcode project, perform the following actions:

1. Start Xcode on the Mac.
2. Select **Xcode** > **Preferences**.



3. In the **Preferences** window, select your account.
   a) Select the **Accounts** tab.
   b) Choose your **Apple ID**.
   c) Choose your **Team**.
   d) Click **View Details**.



4. Access the **Apple Member Center** and retrieve your development team.
5. In a terminal, navigate to `~/.silk/silktest/conf/`.
6. Rename the xcconfig file template `silktest.xcconfig.sample` to `silktest.xcconfig`.
7. Add your development team to the `silktest.xcconfig` file.

   ```
   DEVELOPMENT_TEAM = <your development team>
   ```

8. Execute the following commands in a terminal on the Mac to verify that you have prepared the `WebDriverAgentRunner` project correctly:
   a) Determine the unique device id (udid) of your physical iOS device:

      ```
      idevice_id -l
      ```

   b) Navigate to the `WebDriverAgentRunner` project:

      ```
      cd /Application/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver/WebDriverAgent
      ```

c) Test that the `WebDriverAgent` can be built:

```
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner
–xcconfig ~/.silk/silktest/conf/silktest.xcconfig -destination
'id=<udid>' test
```

Replace the <udid> with the unique device id that you have determined previously.

Tip: If the `xcodebuild` command fails, follow the instructions in the error message. Additionally, open the Preferences window of the `WebDriverAgentRunner` project and ensure that the **Automatically manage signing** check box in the **General** tab is not checked.

9. *Optional:* In the `infoservice.properties` file, you can specify the port for the Silk Test Information Service or capabilities which are used during all test runs on the Mac.

   For additional information, see *Editing the Properties of the Silk Test Information Service*.

Using a Personal Team Profile for Testing on Physical iOS Devices

If you have no membership in the Apple Developer Program, you can use a personal team profile to test an application on a physical iOS device:

1. On the Mac, navigate to `/Application/Silk/Mobile/common/Appium/node_modules/appium-xcuitestdriver/WebDriverAgent`.
2. Open `WebDriverAgent.xcodeproj` project in Xcode.
3. From the **TARGETS** list, select the `WebDriverAgentLib` target:
   a) Click the **General** tab.
   b) Select **Automatically manage signing**.
   c) Select your development team.

   The **Signing Certificate** is automatically selected.
4. From the **TARGETS** list, select the `WebDriverAgentRunner` target:
   a) Click the **General** tab.
   b) Select **Automatically manage signing**.
   c) Select your development team.

   The **Signing Certificate** is automatically selected.
5. If Xcode fails to create a provisioning profile for the `WebDriverAgentRunner` target, manually change the bundle id for the target.
   a) Click the **Build Settings** tab.
   b) Change the **Product Bundle Identifier** to something that Xcode accepts.

      For example, if the **Product Bundle Identifier** is *com.facebook.WebDriverAgentRunner*, change it to *io.appium.WebDriverAgentRunner* or *io.borland.WebDriverAgentRunner*.
   c) Click the **General** tab.

   The target should now have a provisioning profile.
6. Save the `WebDriverAgent.xcodeproj` project.
7. To verify that everything works as expected, build the project:

```
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -
destination 'id=<udid>' test IPHONEOS_DEPLOYMENT_TARGET=10.3
```

8. To avoid problems during the reinstallation of the WebDriverAgent apps, permanently install an additional app that uses the same provisioning profile, on the device. For example, install the `IntegrationApp` of the WebDriverAgent Xcode project:
   a) From the **TARGETS** list, select the `IntegrationApp` target.
   b) Click the **General** tab.
   c) Select **Automatically manage signing**.
   d) Select your development team.

9. If Xcode fails to create a provisioning profile for the `IntegrationApp` target, manually change the bundle id for the target in the same way as described above for the `WebDriverAgentRunner` target.
10. After successfully configuring the `IntegrationApp` target, install and run the `IntegrationApp` on the physical iOS device:
    a) Select the target and the iOS device.
    b) Click **Play**.

Although the apps are successfully installed on the device, an error message like the following might appear in the console or the Appium log files:
2017-01-24 09:02:18.358 xcodebuild[30385:339674] Error Domain=com.apple.platform.iphoneos Code=-12 "Unable to launch com.apple.test.WebDriverAgentRunner-Runner" UserInfo={NSLocalizedDescription=Unable to launch com.apple.test.WebDriverAgentRunner-Runner, NSUnderlyingError=0x7fa839cadc60 {Error Domain=DTXMessage Code=1 "(null)" UserInfo={DTXExceptionKey=The operation couldn't be completed. Unable to launch com.apple.test.WebDriverAgentRunner-Runner because it has an invalid code signature, inadequate entitlements or its profile has not been explicitly trusted by the user. : Failed to launch process with bundle identifier 'com.apple.test.WebDriverAgentRunner-Runner'}}} 2017-01-24 09:02:18.358 xcodebuild[30385:339674] Error Domain=IDETestOperationsObserverErrorDomain Code=5 "Early unexpected exit, operation never finished bootstrapping - no restart will be attempted" UserInfo={NSLocalizedDescription=Early unexpected exit, operation never finished bootstrapping - no restart will be attempted} Testing failed: Test target WebDriverAgentRunner encountered an error (Early unexpected exit, operation never finished bootstrapping - no restart will be attempted)
The problem is that the developer is not trusted on the device. If you manually try to run the apps on the device, you will see an **Untrusted Developer** message.

To solve this issue on the device, go to **Settings** > **General** > **Profiles** or **Settings** > **General** > **Device Management**, depending on the device type and the iOS version. Then trust the developer and allow the apps to be run.

Editing the Properties of the Silk Test Information Service

You can use the `infoservice.properties` file to specify the port for the Silk Test Information Service or various capabilities. These capabilities are applied each time Silk Test executes a test on the machine on which the Silk Test Information Service is running.

1. Navigate to the directory in which the `infoservice.properties.sample` file is located.
   For example, on a Mac, navigate to `~/.silk/silktest/conf/`. On a Windows machine, navigate to `C:\ProgramData\Silk\SilkTest\conf`.
2. Rename the file `infoservice.properties.sample` to `infoservice.properties`.
3. Specify a port that is not in use.
   The default port for the Silk Test Information Service is 22901.
4. To specify capabilities, add the following line to the `infoservice.properties` file:
   ```
   customCapabilities=<custom capability_1>;<custom_capability_2>;...
   ```

> **Example: Running an iOS Simulator in a Specified Language**
>
> To always run a specific iOS Simulator on a Mac in the same language, for example Japanese, specify the custom capabilities *language* and *locale*. To do so, add the following line to the `infoservice.properties` file:
>
> ```
> customCapabilities=language=ja;locale=ja_JP
> ```

Uninstalling the Silk Test Information Service from a Mac

To uninstall the Silk Test information service (information service) from a Mac, for example if you no longer want to execute tests against Apple Safari on the Mac:

1. Create a new shell file, for example `uninstallInfoService.sh`.
2. Type the following code into the new file:

```
#!/bin/sh

if launchctl list | grep com.borland.infoservice ; then
  launchctl unload /Library/LaunchAgents/com.borland.infoservice.plist
  echo "unloading Launch Daemon"
fi

if [ -d "/Applications/Silk" ]
then
  sudo rm -rf /Applications/Silk
fi

if [ -f "/Library/LaunchAgents/com.borland.infoservice.plist" ]
then
  sudo rm /Library/LaunchAgents/com.borland.infoservice.plist
fi

if [ -f "/usr/local/bin/ideviceinstaller" ]
then
  sudo rm /usr/local/bin/ideviceinstaller
fi

exit 0
```

3. In the command line, type `chmod +x uninstallInfoService.sh` to make the shell file executable.
4. Execute the shell file from the command line.

Recommended Settings for iOS Devices

To optimize testing with Silk Test Workbench, configure the following settings on the iOS device that you want to test:

- To make the testing reflect the actions an actual user would perform, disable AutoFill and remembering passwords for Apple Safari. Tap **Settings** > **Safari** > **Passwords & AutoFill** and turn off the **Names and Passwords** setting.
- The iOS device must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings** > **General** > **Passcode Lock**.

Running Existing Scripts on iOS Using XCUITest

Attention: Prior Silk Test Workbench versions used *Instruments* to automate iOS devices. With iOS 9.3, Apple has replaced the support for Instruments with support for the *XCUITest* framework, causing Silk Test Workbench to also no longer support *Instruments*. Because of this change, existing iOS test scripts might break when executed from the current version of Silk Test Workbench.

- The behavior of the `classname` attribute in XCUITest is different to the behavior in Instruments. In most cases, Silk Test Workbench will automatically handle this change. However, if an existing test script breaks because of such a `classname` attribute, you will have to record a new locator for the corresponding object.
- The object hierarchy has changed.

Testing an Installed App

To test a native mobile app that is already installed on a device, an Emulator, or a Simulator, specify the app in the connection string.

1. Open an existing project that tests a native mobile app.

2. Open the **Edit Application Configuration** dialog box.
3. Replace the existing app in the connection string with one of the following:
    - If you want to test an iOS app, replace the app with the *bundleId*, for example replace `app=MyApp.ipa` with `bundleId=silktest.InsuranceMobile`.
    - If you want to test an Android app, replace the app with the *appActivity* and the *appPackage*. For example, replace `app=MyApp.apk` with `appActivity=.LoginActivity;appPackage=silktest.insurancemobile`.

For additional information, see *Connection String*.

Recording Mobile Applications

Once you have established the connection between Silk Test Workbench and a mobile device or an emulator, you can record the actions that are performed on the device. To record mobile applications, Silk Test Workbench uses a **Recording** window that provides the following functionality:

- Displays the screen of the mobile device or Android emulator which you are testing.
- When you perform an action in the **Recording** window, the same action is performed on the mobile device.
- When you interact with a control on the screen, the **Recording** window preselects the default action.
    - If the default action is a `Click`, and you left-click on the control, the action is executed. You can perform a right-click to show a list of the available actions against the control. You can then select the action that you want to perform and click **OK**.
    - If the default action is not a `Click`, a list of all the available actions against the control displays, and you can select the action that you want to perform or simply accept the preselected action by clicking **OK**.

    When you have selected an action from the list, you can type values for the parameters of the selected action into the parameter fields. Silk Test Workbench automatically validates the parameters.
- During recording, Silk Test Workbench displays the mouse position next to the recording window. You can toggle the location to switch between displaying the absolute mouse position on the device display and the mouse position in relation to the active object.
- When you pause the recording, you can perform actions in the screen which are not recorded to bring the device into a state from which you want to continue recording.
- When you stop recording, a script is generated with your recorded actions, and you can proceed with replaying the test.

Selecting the Mobile Device for Test Replay

You can define the mobile device that is used for the replay of a test in the following ways:

- If you execute a test from the UI of Silk Test Workbench and the **Select Mobile Device** dialog box displays, the mobile device, Android Emulator, or iOS Simulator that is selected in the dialog box is used, and Silk Test Workbench ignores which mobile device is set in the test script.
- If the **Select Mobile Device** dialog box is disabled, because the **Don't show again** check box is checked, the application configurations in the individual test scripts determine the mobile device that is used to execute the tests.

    Note: To re-enable the **Select Mobile Device** dialog box, click **Tools** > **Options** and set the **Show Mobile Configuration Dialog** option to **Yes**.
- If you execute a script from the command line or from a Continuous Integration (CI) server, specify the connection string in the application configuration of the script.

    To overwrite the mobile device that is specified in the application configuration, use the `mobiledevice` parameter. For additional information, see the *Parameters* topic of the STW.EXE command line.

- If you execute a test from Silk Central, specify the mobile device in the **Mobile Device Selection** area of the **Deployment** tab of the execution definition in Silk Central instead of specifying a connection string. For additional information, refer to the *Silk Central Help*.

You can use the connection string to specify a specific mobile device, or you can filter a subset of the available devices, for example if you have a device pool. The first matching device is used for replay. If not specified otherwise, mobile devices are matched by using the following rules, with declining priority:

- Matching mobile devices connected to the local machine are preferred over mobile devices connected to remote locations.
- If the browser type is specified in the connection string, newer browser versions are preferred over older versions.
- Newer platforms are preferred over older platforms.
- A physical device is preferred to an Emulator or Simulator.
- A device with a device name that is alphabetically later is preferred. For example, a device named "iphone 6" is preferred to a device named "iphone 5".

Example: Connection string for an app on an Android device that is connected to a remote machine

To test the app `MyApp.apk` on an Android device that is connected to a remote machine, the connection string would look like the following:

```
"platformName=Android;deviceName=MotoG3;host=http://
10.0.0.1;app=MyApp.apk"
```

Example: Connection string for an app on an iOS Simulator on a Mac

```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=1
0.0.0.1;app=MyApp.ipa;isSimulator=true"
```

Using Devices from Mobile Center

Mobile Center is a mobility gateway that enables you to manage the testing of your mobile devices.

To access the devices that are managed through the Mobile Center from Silk Test Workbench, perform the following actions:

1. Integrate Silk Test Workbench with Silk Central.

   For additional information, see *Integrating Silk Test Workbench with Silk Central*.
2. Configure Silk Central to use Mobile Center.

   Note: While installing Mobile Center, ensure that the appropriate Android SDK version is used. Ensure that the same version is used in Silk Test Workbench by setting the environment variable SILK_ANDROID_HOME, for example to `C:\Users\<user>\AppData\Local\Android\android-sdk`. For additional information, refer to the *Silk Central Help*.
3. To test on iOS, ensure that the following IPA files are signed:

   - HP4M-Agent.ipa
   - HPMC-AgentLauncher.ipa
   - WebDriverAgentRunner-Runner.ipa

   Note: Silk Test Workbench does not support testing iOS simulators through Mobile Center.

In the **Select Applications** dialog, you can now select the Mobile Center device on which you want to test.

Note: You cannot test a mobile device with both Silk Test Workbench and Mobile Center at the same time. Restart a mobile device that you have tested with Silk Test Workbench, if you want to continue testing the device from Mobile Center.

Note: When testing on a device that is managed through the Mobile Center, Silk Test Workbench does not support using the methods `TypeKeys` or `SetText` to type key codes like **ENTER**. Additionally, Silk Test Workbench does not support pressing the **Home** button on iOS devices.

Note: When testing on an Android Emulator, disable the GPU HW Acceleration.

Using SauceLabs Devices

SauceLabs provides an automated testing platform, enabling you to test on various mobile devices and mobile platform versions without having to purchase and maintain your own infrastructure.

To access SauceLabs devices through Silk Central, perform the following actions:

1. Ensure that Silk Test Workbench is integrated with Silk Central.
   For additional information, see *Integrating Silk Test Workbench with Silk Central*.
2. Ensure that Silk Central is configured to use SauceLabs.
   For additional information, refer to the *Silk Central Help*.

In the **Select Applications** dialog, you can now select the SauceLabs device on which you want to test.

Connection String for a Mobile Device

The *connection string* specifies which mobile device is used for testing. When performing mobile testing, Silk Test Workbench uses the connection string to connect to the mobile device. The connection string is typically part of the application configuration. You can set the connection string when you configure your application under test. To change the connection string, you can use the **Edit Application Configuration** dialog box.

Note: If you execute a test from Silk Central, specify the mobile device in the **Mobile Device Selection** area of the **Deployment** tab of the execution definition in Silk Central instead of specifying a connection string. For additional information, refer to the *Silk Central Help*.

You can use the connection string to specify a specific mobile device, or you can filter a subset of the available devices, for example if you have a device pool. The first matching device is used for replay. If not specified otherwise, mobile devices are matched by using the following rules, with declining priority:

- Matching mobile devices connected to the local machine are preferred over mobile devices connected to remote locations.
- If the browser type is specified in the connection string, newer browser versions are preferred over older versions.
- Newer platforms are preferred over older platforms.
- A physical device is preferred to an Emulator or Simulator.
- A device with a device name that is alphabetically later is preferred. For example, a device named "iphone 6" is preferred to a device named "iphone 5".

The following components are available for the connection string:

Component	Description
deviceName	The name of the mobile device. When testing on a physical mobile device, the device ID can be used instead. Supports wildcards. Case-insensitive.
platformName	Android or iOS. Required.

Component	Description
deviceId	*Optional:* The ID of the mobile device. Can be used instead of the device name, when testing on a physical mobile device. Supports wildcards. Case-insensitive.
platformVersion	*Optional:* The Android or iOS version. Specify the version to test only on mobile devices that have a specific Android or iOS version. Supports wildcards. Case-insensitive.
browserVersion	*Optional:* Can be used in combination with the browser type to test only on the specified browser version. Supports wildcards. Case-insensitive.
host	*Optional:* If not set, any specified remote location can be used as the host. Supports wildcards. Case-insensitive.
• app • appActivity • appPackage	Required for testing native mobile applications on Android. Either the full path to the app, or a combination of *appActivity* and *appPackage*. For example `app=MyApp.apk` or `appActivity=.LoginActivity;appPackage=silktest.insurancemobile`
• app • bundleId	Required for testing native mobile applications on iOS. Either the full path to the app or the *bundleId*. For example `app=MyApp.ipa` or `bundleId=silktest.InsuranceMobile`
noReset	*Optional:* Can be set when testing native mobile applications. Is only valid if the *app* is specified. True if the app should not be reinstalled before testing. False if the app should be reinstalled before testing. The default value is False.
isSimulator	*Optional:* Used to specify that the test should only be executed on an iOS Simulator. The device name can be used instead.
isPhysicalDevice	*Optional:* Used to specify that the test should only be executed on a physical device. The device name can be used instead.

When using a pool of devices and to find out which device is actually used for testing, you can use the return value of the `GenerateConnectionString` method of `MobileDevice` class.

Testing a mobile web application on a mobile device or on an Android Emulator

When testing a mobile web application on a mobile device or on an Android Emulator, the connection string consists of the following parts:

1. The mobile device name, for example MotoG3, or the device ID, for example 11111111.

   Note: If the device name is unique, Micro Focus recommends to use the device name in the connection string, because the device ID is less readable.
2. The platform name.
3. The browser version. This can only be used in combination with setting the browser type
4. The IP address or the host name of a specific remote machine, for example 10.0.0.1. You can also use the name of a remote location that is specified in the **Edit Remote Locations** dialog box as the host name, for example *MyRemoteLocation*. When using the remote location name, you can also use wildcards. To test an Android device that is connected to the local machine, specify the IP address or the host name of the local machine.

Example: Connection string for any available Android device

```
"platformName=Android"
```

Example: Connection string for a browser on an Android device that is connected to the local machine

To test a mobile browser on an Android device that is connected to the local machine, the connection string should look similar to the following:

```
"deviceName=MotoG3;platformName=Android;host=localhost"
```

or

```
"platformName=Android;deviceId=11111111;host=localhost"
```

Example: Connection string for a browser on an Android device that is connected to a remote machine

To test a mobile browser on a remote Android device, the connection string should look similar to the following:

```
"deviceName=MotoG3;platformName=Android;host=10.0.0.1"
```

```
"deviceName=MotoG3;platformName=Android;host=MyRemoteLocation*"
```

Example: Connection string for a browser on an iOS device that is connected to a Mac

To test a mobile browser on a remote iOS device, the connection string would look like the following:

```
"deviceName=myiPhone6;platformName=iOS;host=10.0.0.1"
```

Testing a native mobile application on a mobile device or on an Android Emulator

When testing a native mobile application on a mobile device or on an Android Emulator, the connection string consists of the following parts:

1. The mobile device name, for example MotoG3, or the device ID, for example 11111111.

   Note: If the device name is unique, Micro Focus recommends to use the device name in the connection string, because the device ID is less readable.
2. The platform name.
3. The IP address or the host name of a specific remote machine, for example 10.0.0.1. You can also use the name of a remote location that is specified in the **Edit Remote Locations** dialog box as the host name, for example *MyRemoteLocation*. When using the remote location name, you can also use wildcards. To test an Android device that is connected to the local machine, specify the IP address or the host name of the local machine.
4. The name of the file of the app that you want to test, or the URL of the file, if the file is located on a web server. For example `C:/MyApp.apk` or `MyApp.ipa`.

   - Android apps are always `.apk` files.
   - iOS apps on a real device are always .ipa files.
   - iOS apps on a Simulator are either a zipped file or a directory with the name *app*.

Example: Connection string for an app on an Android device that is connected to a remote machine

To test the app `MyApp.apk` on an Android device that is connected to a remote machine, the connection string would look like the following:

```
"platformName=Android;deviceName=MotoG3;host=http://
10.0.0.1;app=MyApp.apk"
```

Example: Connection string for an app on an iOS device that is connected to a Mac

To test the app `MyApp.ipa` on an iOS device that is connected to a remote machine, the connection string would look like the following:

```
"platformName=iOS;deviceName=MyiPhone;host=http://
10.0.0.1;app=MyApp.ipa"
```

Testing a mobile web application on an iOS Simulator

When testing a mobile web application on an iOS Simulator, the connection string consists of the following parts:

1. The platform name, which is iOS.
2. The platform version, for example 10.0.
3. The mobile device name, for example iPhone6.
4. The IP address or the host name of the Mac, on which the iOS Simulator is running.

Example: Connection string for a browser on an iOS Simulator on a Mac

```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=1
0.0.0.1;isSimulator=true"
```

Testing a native mobile application on an iOS Simulator

When testing a native mobile application on an iOS Simulator on a Mac, the connection string consists of the following parts:

1. The platform name, which is iOS.
2. The platform version, for example 10.0.
3. The mobile device name, for example iPhone6.
4. The IP address or the host name of the remote machine, for example 10.0.0.1.
5. The name of the app that you want to test, for example `MyApp.ipa`.

Example: Connection string for an app on an iOS Simulator on a Mac

```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=1
0.0.0.1;app=MyApp.ipa;isSimulator=true"
```

Interacting with a Mobile Device

To interact with a mobile device and to perform an action like a swipe in the application under test:

1. In the **Recording** window, click **Show Mobile Device Actions**. All the actions that you can perform against the mobile device are listed.
2. Select the action that you want to perform from the list.
3. To record a swipe on an Android device or emulator, move the mouse while holding down the left mouse button.
4. Continue with the recording of your test.

Releasing a Mobile Device

When recording or playing back a test against a mobile device, the Open Agent instance takes ownership of the device. By doing so, the Open Agent is preventing other Silk Test users from using the device. To

enable other Silk Test users to use the device after you have finished recording or replaying tests on the device, Silk Test automatically releases the device when the Silk Test client is closed, when an unattended test process finishes, or when the Open Agent is closed. You can also manually release the device.

Note: Releasing a mobile device will close the application under test (AUT) on the mobile device.

Releasing a Mobile Device After Recording

Release a mobile device after recording to enable other Silk Test users to test on the device.

To release a mobile device after you have finished recording, perform one of the following actions:

- Stop the Open Agent from the System Tray.
- Close Silk Test Workbench. The device is only released by this action when parallel testing is enabled.

Note: Releasing a mobile device will close the application under test (AUT) on the mobile device.

Releasing a Mobile Device After Replay

Release a mobile device after replay to enable other Silk Test users to test on the device.

To manually release a mobile device after replaying is complete, you can also perform one of the following:

- If you have tested a mobile web application, use the `Close` method or the `CloseSynchron` method of the `BrowserApplication` class. For additional information on these methods, refer to the API documentation.

  ```
  'VB .NET code
  WebBrowser.Close()
  ```
- If you have tested a mobile native application, use the `CloseApp` method of the `MobileDevice` class.

  For example, type the following:

  ```
  'VB .NET code
  Dim mobileDevice = _desktop.MobileDevice()
  mobileDevice.CloseApp()
  ```
- Add the *Agent*.`DetachAll`() statement to the test script, to release the device after replaying VB .NET tests.

A mobile device is automatically released if one of the following conditions is met:

- The Open Agent is closed.
- The test process stops during unattended testing. The device is only released by this action when parallel testing is enabled.
- Silk Test Workbench is closed. The device is only released by this action when parallel testing is enabled.

Note: Releasing a mobile device will close the application under test (AUT) on the mobile device.

Using Devices Managed By Silk Central

To manage your mobile tests and to perform configuration testing, you can use Silk Central.

Integrate Silk Test Workbench with Silk Central.

For information on integrating Silk Test Workbench with Silk Central, see *Integrating Silk Test Workbench with Silk Central*.

Any device providers configured for the specified user in Silk Central are displayed in the list of remote browsers and mobile devices in the **Select Application**, **Select Browser**, and **Select Mobile Device**

dialog boxes. Silk Test Workbench displays any devices that are managed by Silk Central with the name and the host name/URL of the device provider that is configured in Silk Central.

Troubleshooting when Testing Mobile Applications

Why does the Select Application dialog not display my mobile devices?

If Silk Test Workbench does not recognize a mobile device or emulator, the **Mobile** tab in the **Select Application** dialog does not display the device or emulator. Additionally, the **Web** tab of the **Select Application** dialog does not display the mobile browsers that are installed on the device or emulator.

Silk Test Workbench might not recognize a mobile device or emulator for one of the following reasons:

Reason	Solution
The emulator is not running.	Start the emulator.
The Android Debug Bridge (adb) does not recognize the mobile device.	To check if the mobile device is recognized by adb: **1.** Navigate to the Android Debug Bridge (adb) in the Android SDK installation folder. If the Android SDK is not installed, navigate to `C:\Program Files (x86)\Silk\SilkTest\ng\Mobile\windows\AndroidTools\platform-tools` to use the adb that is installed with Silk Test Workbench. **2.** Hold **Shift** and right-click into the **File Explorer** window. **3.** Select **Open command window here**. **4.** In the command window, type `adb devices` to get a list of all attached devices. **5.** If your device is not listed, check if USB-debugging is enabled on the device and if the appropriate USB driver is installed. **6.** If you get an error, for example `adb server is out of date`, ensure that the adb version in `C:\Program Files (x86)\Silk\SilkTest\ng\Mobile\windows\AndroidTools\platform-tools` is the same as the adb version of your local Android SDK. For additional information, see *What can I do if the connection between the Open Agent and my device is unstable?*.
The version of the operating system of the device is not supported by Silk Test Workbench.	For information on the supported mobile operating system versions, refer to the *Release Notes*.
The USB driver for the device is not installed on the local machine.	Install the USB driver for the device on the local machine. For additional information, see *Installing a USB Driver*.
USB-debugging is not enabled on the device.	Enable USB-debugging on the device. For additional information, see *Enabling USB-Debugging*.

Note: If all previous solutions do not work, you could try to restart the device.

Why does Silk Test Workbench search for a URL in Chrome for Android instead of navigating to the URL?

Chrome for Android might in some cases interpret typing an URL into the address bar as a search. As a workaround you can manually add a command to your script to navigate to the URL.

What do I do if the adb server does not start correctly?

When the Android Debug Bridge (adb) server starts, it binds to local TCP port 5037 and listens for commands sent from adb clients. All adb clients use port 5037 to communicate with the adb server. The adb server locates emulator and device instances by scanning odd-numbered ports in the range 5555 to 5585, which is the range used by emulators and devices. Adb does not allow changing those ports. If you encounter a problem while starting adb, check if one of the ports in this range is already in use by another program.

For additional information, see *http://developer.android.com/tools/help/adb.html*.

What can I do if the connection between the Open Agent and my device is unstable?

If you have installed the Android SDK or another tool that uses the Android Debug Bridge (adb), an additional adb server might be running in addition to the one that is used by Silk Test Workbench. If the running adb servers have different versions, the connection between the Open Agent and the device might become unstable or even break.

To avoid version mismatch errors, specify the path to the Android SDK directory by setting the environment variable *SILK_ANDROID_HOME*, for example to `C:\Users\<user>\AppData\Local\Android\android-sdk`. If the information service was running during this change, use the Windows Service Manager to restart the Silk Test information service with the updated environment variable. If the variable is not set, Silk Test Workbench uses the adb version that is shipped with Silk Test Workbench.

Why do I get the error: `Failed to allocate memory: 8`?

This error displays if you are trying to start up the emulator and the system cannot allocate enough memory. You can try the following:

1. Lower the RAM size in the memory options of the emulator.
2. Lower the RAM size of Intel HAXM. To lower the RAM size, run the `IntelHaxm.exe` again and choose **change**.
3. Open the **Task Manager** and check if there is enough free memory available. If not, try to free up additional memory by closing a few programs.

Why do I get the error "Silk Test cannot start the app that you have specified" during testing on an iOS device?

This error might display for one or more of the following reasons:

Reason	Solution
The iOS device is not in developer mode.	You can enable the developer mode in one of the following two ways: • Connect the device to a Mac on which Xcode is installed, and start the app that you want to test on the device. • Add your provisioning profiles to the device. 1. Open Xcode. 2. Select **Window** > **Devices**. 3. Right-click on the iOS device.

Reason	Solution
	4. Select **Show Provisioning Profiles**. 5. Add your provisioning profiles.
You have recently updated the iOS version of the device.	1. Open Xcode. 2. Select **Window** > **Devices**. 3. Wait unitl Xcode has processed the symbol files.
UI automation is not enabled on the iOS device.	1. Select **Settings** > **Developer**. 2. Activate **Enable UI Automation**.
The **Web Inspector** is not activated on the iOS device, while you are trying to test a mobile web application.	1. Click **Settings** > **Safari** > **Advanced**. 2. Activate the **Web Inspector**.
The app that you want to test was not built for the iOS version of the iOS device on which you are testing.	Use Xcode to build the app for the iOS version of the device.
The **Software Update** dialog box is currently open on the iOS device.	Close the dialog box and disable automatic software updates: 1. Select **Settings** > **App and iTunes Stores** > **AUTOMATIC DOWNLOADS**. 2. Deactivate **Updates**.

Why does my Android device display only the Back button in the dynamic hardware controls?

If the Android or the Android Emulator is screen-locked when you start testing, the device or Emulator might display only the button **Back** in the dynamic hardware controls.

To solve this issue, stop the Open Agent, restart the device, and change the device settings to no longer lock the screen.

Why does my Android device or emulator no longer display a keyboard?

To support unicode characters, Silk Test Workbench replaces the standard keyboard with a custom keyboard. When testing is finished, the original keyboard is restored. If an error occurs during testing, the custom keyboard might still be active and cannot be replaced.

To solve this issue, manually reset the keyboard under **Settings** > **Language & input** > **Current Keyboard**.

Why does my device not respond during testing?

If the device, emulator, or Simulator is screen-locked when you start testing, and Silk Test Workbench is unable to unlock the screen, the device, emulator, or Simulator might stop responding to any actions.

To solve this issue, stop the Open Agent and change the device settings to no longer lock the screen.

Why can I not install the Information Service on a Mac?

When the **Allow apps downloaded from** setting in the **General** tab of the **Security & Privacy** system preferences pane is set to **Mac App Store and identified developers**, which is the default value, the following error message appears when opening the Information Service setup:
"SilkTestInformationService<version>.pkg" can't be opened because it is from an unidentified developer.

To solve this issue, use one of the following:

- Right-click the setup file and select **Open**. A warning message will appear, but you will still be able to open the file.
- Set the **Allow apps downloaded from** setting to **Anywhere**.
- After attempting to open the file, navigate to the **General** tab of the **Security & Privacy** system preferences pane and click **Open Anyway**.

Why is the Recording window black when recording an Android app?

Android apps that require a higher level of security, for example apps that handle financial transactions, might have the *FLAG_SECURE* flag set, which prevents Silk Test Workbench from capturing the app. Silk Test Workbench relies on screenshots or on a video of the Android device during recording and will display a black screen of the device in the **Recording** window, if the Android app that you are testing has this flag set. To test such an app with Silk Test, you have to contact the app development team, and ask them to un-set the FLAG_SECURE flag during testing.

Why does Silk Test Workbench not show a video when testing on an Android emulator?

If the emulator is using the graphic card of your computer for better rendering, the video capturing of Silk Test Workbench might not work. To solve this, emulate the graphics in software:

1. Open the **Android Virtual Device Manager**.
2. Click **Edit** in the **Actions** column of the emulator.
3. Select **Software** from the list in the **Emulated Performance** area of the **Virtual Device Configuration** dialog.

What can I do if Silk Test Workbench does not show a video when testing in a cloud environment?

When testing in a cloud environment, showing a video might not work when recording or replaying a test, for example because required ports are not open.

To solve this issue, you can specify a list of WebDriver host URLs in the `infoservice.properties` file. For information on how to access this properties file, see *Editing the Properties of the Silk Test Information Service*. Add the option infoservice.disableScreencastHosts to the file, by typing the following:

```
infoservice.disableScreencastHosts=<URL_1>,<URL_2>, ...
```

For example:

```
infoservice.disableScreencastHosts=http://my-webdriver-server-url.com:
80/wd/hub
```

You can specify URL patterns like **my-webdriver-server-url.com* by using asterisks (*) as wildcards.

Silk Test Workbench will show a series of screenshots instead of a video when recording and replaying on the specified hosts.

How can I change the installed version of Xcode?

If the version of Xcode that you are using is not supported by Silk Test Workbench, for example when you upgrade to the latest version of Xcode, an error message might appear when testing on iOS.

To replace the installed version of Xcode with a supported version, download a supported Xcode version from *https://developer.apple.com/download/more/*, and replace the unsupported version with the downloaded version. For information about the supported Xcode versions, refer to the *Release Notes*.

What can I do if my Mac runs out of disk space?

Silk Test Workbench uses Instruments to automate iOS devices. This tool creates large log files in the `/Library/Caches/com.apple.dt.instruments` directory, which might fill up disk space on the Mac. To solve this issue, Micro Focus recommends regularly deleting these log files, either manually or by using a cronjob. For example, to delete the files each day at the same time, you could do the following:

1. Type `sudo crontab -e` into a Terminal. This opens an editor in which you can edit the crontab for root.
2. Add the following line to the crontab:
   ```
   0 2 1 * * find /Library/Caches/com.apple.dt.instruments -mtime +10 -delete
   ```
3. Save the crontab.

In this example, all log files that are older than ten days will be deleted each day at 2 AM from the directory.

Why does my test fail with the error message "Unable to sign WebDriver Agent for testing "?

When testing on a physical iOS device, this error usually means that during the build process the WebDriverAgent app could not be signed or that there is a problem with the provisioning profile.

You can check the actual problem with the following commands, which have to be executed at the Mac machine to which the device is connected:

```
cd /Applications/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -
destination 'id=<udid>' test
```

Verify that the folder `Resources` exists under `/Applications/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver` and that the folder contains the file `WebDriverAgent.bundle`. If not, create this folder and an empty `WebDriverAgent.bundle` file, for example by using the following command:

```
mkdir -p Resources/WebDriverAgent.bundle
```

What can I do to prevent the Developer Tools Access from requesting to take control of another process?

When starting the execution of a test on iOS, a message box stating the following might appear: Developer Tools Access needs to take control of another process for debugging to continue. Type your password to allow this.

To avoid getting this message, execute the following command in a Terminal:

```
sudo /usr/sbin/DevToolsSecurity --enable
```

Why are the rectangles wrong while testing a mobile web application on an iPad?

If the rectangles around controls are offset when testing a mobile web application on an iPad, you might have multiple browser tabs open and the Tab bar might be displayed. To fix this issue, close all tabs except one.

Why can I no longer record or replay tests on my device after updating Silk Test Workbench?

When updating to a new version of Silk Test Workbench, some Appium apps on any physical mobile devices that have already been used for mobile testing with the previous version of Silk Test Workbench are updated automatically. If for any reason these apps are not automatically updated, you might experience difficulties when trying to record or replay tests on the device.

If you are experiencing such issues on a specific Android device after updating Silk Test Workbench, manually uninstall the following apps from the device:

- Appium Android Input Manager
- Appium Settings
- io.appium.uiautomator2.server
- io.appium.uiautomator2.server.test
- Unlock

If you are experiencing such issues on a specific iOS device after updating Silk Test Workbench, manually uninstall the WebDriverAgentRunner from the device.

Why can I not record a mobile application?

Silk Test Workbench uses Appium to test mobile applications. Some network proxy settings set in Appium might interfere with recording Silk Test Workbench. You could try to deactivate the network proxy settings on the mobile device or Emulator.

How Can I Use Chrome for Android to Replay Tests?

By default you can use the **Select Browser** dialog box to select the browser during replay.

If you execute a script from the command line or from a Continuous Integration (CI) server, you can specify the connection string in the application configuration of the script. To overwrite the browser that is specified in the application configuration, use the *silktest.configurationName* environment variable.

You can also use the property `browsertype` of the `BrowserApplication` class to set the type of the browser that is used during replay. However, the `browsertype` does not include an explicit value for Chrome for Android.

To specify that you want to use Chrome for Android as the browser, on which a test is replayed, set the `browsertype` to *GoogleChrome* and specify Android as the platform. When Android is specified, Silk Test Workbench uses Chrome for Android instead of Google Chrome on a desktop machine to execute the test.

Examples

The following code sample shows how you can set the base state for a test to use Chrome for Android on a Nexus 7 by using the *silktest.configurationName*:

```
SET
silktest.configurationName="platformName=Android;deviceName=Nexu
s 7;host=10.0.0.1 - Chrome"
```

The following code sample shows how you can set the base state for a test to use Chrome for Android by using the `browsertype` :

```
'VB .NET code
Dim baseState = New BrowserBaseState(BrowserType.GoogleChrome,
"demo.borland.com/InsuranceWebExtJS/")
baseState.SetConnectionString("platformName=Android")
baseState.Execute()
```

Limitations for Testing Mobile Web Applications

The support for playing back tests and recording locators on mobile browsers is not as complete as the support for the other supported browsers. The known limitations for playing back tests and recording locators for mobile web applications are:

- The following classes, interfaces, methods, and properties are currently not supported for mobile web applications:
  - `BrowserApplication` class.
    - `CloseOtherTabs` method
    - `CloseTab` method
    - `ExistsTab` method
    - `GetActiveTab` method
    - `GetSelectedTab` method
    - `GetSelectedTabIndex` method
    - `GetSelectedTabName` method
    - `GetTabCount` method

    - `ImageClick` method
    - `OpenTab` method
    - `SelectTab` method
  - `DomElement` class.
    - `DomDoubleClick` method
    - `DomMouseMove` method
    - `GetDomAttributeList` method
  - `IKeyable` interface.
    - `PressKeys` method
    - `ReleaseKeys` method
- Silk Test Workbench does not support testing HTML frames and iFrames with Apple Safari on iOS.
- Recording in landscape mode is not supported for emulators that include virtual buttons in the system bar. Such emulators do not correctly detect rotation and render the system bar in landscape mode to the right of the screen, instead of the lower part of the screen. However, you can record against such an emulator in portrait mode.
- Only HTML attributes in the HTML DOM are supported in XPath expressions for mobile applications. Silk Test Workbench does not support properties in XPath expressions.
- If you are testing a mobile web application on Android, Silk Test Workbench does not support zooming.
- The following JavaScript alert-handling methods of the `BrowserWindow` class do not work when testing on the Original Android Stock (AOSP) Browser:
  - `AcceptAlert` method
  - `DismissAlert` method
  - `GetAlertText` method
  - `IsAlertPresent` method
- At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac.
- Before starting to test a mobile web application, ensure that no browser tab is open.

  Tip: On iPads you can disable tabs in Apple Safari. Navigate to **Settings** > **Safari** and disable **Show Tab Bar** to do so.
- While testing a mobile web application, you can only have one browser tab open.
- Silk Test Workbench does not support testing mobile web applications that are opened by a native mobile application.

Limitations for Testing Native Mobile Applications

The known limitations for playing back tests and recording locators on native mobile applications (apps) are:

- The following classes, interfaces, methods, and properties are currently not supported for native mobile applications:
  - `IKeyable` interface.
    - `PressKeys` method
    - `ReleaseKeys` method
  - `MobileDevice` class.
    - When testing a native mobile application on iOS, the `SetLocation` method is not supported.
    - When testing a native mobile application on an Android version prior to Android 6.0, you have to enable **Allow mock locations** to use the `SetLocation` method. To do so, open the settings of the Android device or emulator and tap **Developer Options**.

    - When testing a native mobile application on Android 6.0 or later, you have to set the app to **Appium Settings** to use the `SetLocation` method. To do so, open the settings of the Android device or emulator and tap **Developer Options** > **Select mock location app**. Then choose **Appium Settings**.

      Note: The **Appium Settings** entry is only available if you have already executed a test with Appium on the Android device or emulator.
    - When testing on iOS, the `Find` method does not work in the following cases:
      - If the `path` attribute is set.
      - If an attribute value is empty.
- When testing on iOS, the `getValue` method of the `XCUIElementTypeSwitch` class returns the strings `false` or `true` depending on the checked state, instead of returning the strings `0` and `1`.
- Recording in landscape mode is not supported for Android emulators that include virtual buttons in the system bar. Such emulators do not correctly detect rotation and render the system bar in landscape mode to the right of the screen, instead of the lower part of the screen. However, you can record against such an emulator in portrait mode.
- Only HTML attributes in the HTML DOM are supported in XPath expressions for mobile applications. Silk Test Workbench does not support properties in XPath expressions.
- At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac. With Silk Test 17.5 Hotfix 1 or later, you are no longer required to use multiple user sessions on a Mac to test mobile applications on iOS.
- Silk Test Workbench does not support text recognition when testing native mobile applications on both Android and iOS.

  Text recognition includes the following methods:

  - `TextCapture`
  - `TextClick`
  - `TextExists`
  - `TextRectangle`
- Silk Test Workbench does not support testing native mobile applications with multiple web views.
- When testing on iOS, the state of the `isVisible` property is always true, even if the element is not visible.
- When testing on iOS, a swipe action with multiple steps swipes to a point, releases the mouse pointer and then swipes to the next point. On prior versions of iOS, the action does not release the mouse pointer between the swipes.
- When testing on iOS, Silk Test Workbench does not support any multi-touch actions except pinch.
- When testing on iOS, Silk Test Workbench does not support the `PinchIn` method.
- When testing on iOS, you can only accept or dismiss alert dialog boxes. If no **Cancel** button is available and Silk Test Workbench cannot dismiss the dialog, the default action is to accept the dialog.
- When testing on Android, Silk Test Workbench does not provide automated synchronization for controls of the *Animation* class.
- When testing toasts on Android, the following limitations apply:
  - During recording, Silk Test Workbench always displays the rectangle for the toast in the lowest quarter of the **Recording** window, independent of the actual position of the toast.
  - During recording and replay, the detection of a toast by Silk Test Workbench always has a duration of five seconds, even if the toast appears in a shorter time period.
- When testing on iOS, Silk Test Workbench does not provide automated synchronization for controls that call the `UIView.animate` function or the `UIView.animateWithDuration` function.

  You can workaround this issue by increasing the speed of the animation in the app delegate:

```
func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {
  //...
  if NSProcessInfo.processInfo().environment["automationName"] == "Silk
```

```
Test" {
    // Speed animations up (recommended)
    window!.layer.speed = 100;
  }
}
```

Micro Focus does not recommend disabling such animations completely, as this might change the applications behavior. However, if speeding up the animation does not resolve the synchronization issue, you could completely disable animations in the app delegate as follows:

```
func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {
  //...
  if NSProcessInfo.processInfo().environment["automationName"] == "Silk
Test" {
    UIView.setAnimationsEnabled(false)
  }
}
```

- When testing on iOS, the following additional limitations apply:
  - You might experience performance decreases while recording and replaying tests.
  - Due to internal changes in iOS, the locators of some controls might have changed, and some of your existing tests might break.
  - Text fields that are not in focus might not be recognized as text fields. To ensure that text fields are recognized correctly, set the focus on the text fields, for example by clicking on a text field before trying to interact with it.

Clicking on Objects in a Mobile Website

When clicking on an object during the recording and replay of an automated test, a mobile website presents the following challenges in comparison to a desktop website:

- Varying zoom factors and device pixel ratios.
- Varying screen sizes for different mobile devices.
- Varying font and graphic sizes between mobile devices, usually smaller in comparison to a website in a desktop browser.
- Varying pixel size and resolution for different mobile devices.

Silk Test Workbench enables you to surpass these challenges and to click the appropriate object on a mobile website.

When recording a test on a mobile device, Silk Test Workbench does not record coordinates when recording a `Click`. However, for cross-browser testing, coordinates are allowed during replay. You can also manually add coordinates to a `Click`. Silk Test Workbench interprets these coordinates as the HTML coordinates of the object. To click on the appropriate object inside the `BrowserWindow`, during the replay of a test on a mobile device, Silk Test Workbench applies the current zoom factor to the HTML coordinates of the object. The device pixel coordinates are the HTML coordinates of the object, multiplied with the current zoom factor.

If the object is not visible in the currently displayed section of the mobile website, Silk Test Workbench scrolls to the appropriate location in the website.

Example

The following code shows how you can test a `DomButton` with a fixed size of 100 x 20 px in your HTML page.

VB

```
DomButton domButton = _desktop.Find(Of DomButton)("locator for
the button");
domButton.Click(MouseButton.LEFT, new Point(50, 10));
```

During replay on a different mobile device or with a different zoom factor, the `DomButton` might for example have an actual width of 10px on the device screen. Silk Test Workbench clicks in the middle of the element when using the code above, independent of the current zoom factor, because Silk Test Workbench interprets the coordinates as HTML coordinates and applies the current zoom factor.

Using Existing Mobile Web Tests

Silk Test 17.0 or later uses a different approach to mobile web testing than previous versions of Silk Test. This change might result in your old mobile web tests no longer working on Silk Test 17.0 or later. This topic describes some of the changes that were introduced with Silk Test 17.0 and provides guidance on changing existing mobile web tests with Silk Test 17.0 or later.

The following changes for mobile web testing were introduced with Silk Test 17.0:

- With previous versions of Silk Test, you were able to test on iOS devices that were connected by USB to a Windows machine. With Silk Test 17.0 or later, you can only test on iOS devices that are connected to an OSX machine (Mac).
- If you have tested mobile web applications on an Android device with a previous version of Silk Test, you have to manually remove the proxy from the Android device to test a web application with Silk Test 17.0 or later. Silk Test 17.0 or later no longer requires a proxy, and if the proxy is set, the message `Unable to connect to the proxy server` displays on the device.

.NET Support

Silk Test provides built-in support for testing .NET applications including:

- Windows Forms (Win Forms) applications
- Windows Presentation Foundation (WPF) applications
- Microsoft Silverlight applications

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Windows Forms Support

Silk Test Workbench provides built-in support for testing .NET standalone and No-Touch Windows Forms (Win Forms) applications. However, side-by-side execution is supported only on standalone applications.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Object Recognition

The name that was given to an element in the application is used as `automationId` attribute for the locator if available. As a result, most objects can be uniquely identified using only this attribute.

Supported Controls

For a complete list of the record and replay controls available for Win Forms testing, see *Windows Forms Class Reference*.

Windows Forms Class Reference

When you configure a Windows Forms application, Silk Test Workbench automatically provides built-in support for testing standard Windows Forms controls.

Attributes for Windows Forms Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Windows Forms applications include:

- automationid
- caption
- windowid
- priorlabel (For controls that do not have a caption, the priorlabel is used as the caption automatically. For controls with a caption, it may be easier to use the caption.)

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards `?` and `*`.

Custom Attributes for Windows Forms Applications

Windows Forms applications use the predefined automation property `automationId` to specify a stable identifier for the Windows forms control.

Silk Test Workbench automatically will use this property for identification in the locator. Windows Forms application locators look like the following:

```
/FormsWindow//PushButton[@automationId='btnBasicControls']
```

Dynamically Invoking Windows Forms Methods

Dynamic invoke enables you to directly call methods, retrieve properties, or set properties, on an actual instance of a control in the application under test. You can also call methods and properties that are not available in the Silk Test Workbench API for this control. Dynamic invoke is especially useful when you are working with custom controls, where the required functionality for interacting with the control is not exposed through the Silk Test Workbench API.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

Call dynamic methods on objects with the `Invoke` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Call multiple dynamic methods on objects with the `InvokeMethods` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.

For example, to call a method named `SetTitle`, which requires the title to be set as an input parameter of type string, on an actual instance of a control in the application under test, type the following:

```
control.Invoke("SetTitle", "my new title")
```

Note: Typically, most properties are read-only and cannot be set.

Note: Reflection is used in most technology domains to call methods and retrieve properties.

The Invoke Method

For a Windows Forms or a WPF control, you can use the `Invoke` method to call the following methods:

- Public methods that the MSDN defines for the control.
- Public static methods that the MSDN defines.
- User-defined public static methods of any type.

First Example for the Invoke Method

For an object of the Silk Test Workbench type `DataGrid`, you can call all methods that MSDN defines for the type `System.Windows.Forms.DataGrid`.

To call the method `IsExpanded` of the `System.Windows.Forms.DataGrid` class, use the following code:

```
//VB .NET code
Dim isExpanded As Boolean = dataGrid.Invoke("IsExpanded", 3)
```

Second Example for the Invoke Method

To invoke the static method `String.Compare(String s1, String s2)` inside the AUT, use the following code:

```
//VB .NET code
Dim result as Integer = (Integer)
mainWindow.Invoke("System.String.Compare", "a", "b")
```

Third Example for the Invoke Method

This example shows how you can dynamically invoke the user-generated method `GetContents`.

You can write code which you can use to interact with a control in the application under test (AUT), in this example an `UltraGrid`. Instead of creating complex dynamic invoke calls to retrieve the contents of the `UltraGrid`, you can generate a new method `GetContents` and then just dynamically invoke the new method.

In Visual Studio, the following code in the AUT defines the `GetContents` method as a method of the `UltraGridUtil` class:

```
//C# code, because this is code in the AUT
namespace UltraGridExtensions {
  public class UltraGridUtil {
    /// <summary>
    /// Retrieves the contents of an UltraGrid as nested list
    /// </summary>
    /// <param name="grid"></param>
    /// <returns></returns>
    public static List<List<string>>
GetContents(Infragistics.Win.UltraWinGrid.UltraGrid grid) {
      var result = new List<List<string>>();
      foreach (var row in grid.Rows) {
        var rowContent = new List<string>();
        foreach (var cell in row.Cells) {
          rowContent.Add(cell.Text);
        }
        result.Add(rowContent);
      }
      return result;
    }
  }
}
```

The code for the `UltraGridUtil` class needs to be added to the AUT. You can do this in the following ways:

- The application developer can compile the code for the class into the AUT. The assembly needs to be already loaded.
- You can create a new assembly that is loaded into the AUT during test execution.

To load the assembly, you can use the following code:

```
FormsWindow.LoadAssembly(String assemblyFileName)
```

You can load the assembly by using the full path, for example:

```
mainWindow.LoadAssembly("C:/temp/ultraGridExtensions.dll")
```

You can also find the location of the assembly by using the `Location` method:

```
//VB.NET code
Dim assemblyLocation =
GetType(UltraGridExtensions.UltraGridUtil).Assembly.Location
mainWindow.LoadAssembly(assemblyLocation)
```

When the code for the UltraGridUtil class is in the AUT, you can add the following code to your test script to invoke the `GetContents` method:

```
var contents = (IList)
mainWindow.Invoke("UltraGridExtensions.UltraGridUtil.GetContents
", ultraGrid);
```

The `mainWindow` object, on which the `Invoke` method is called, only identifies the AUT and can be replaced by any other object in the AUT.

The InvokeMethods Method

For a Windows Forms or a WPF control, you can use the `InvokeMethods` method to invoke a sequence of nested methods. You can call the following methods:

- Public methods that the MSDN defines for the control.
- Public static methods that the MSDN defines.
- User-defined public static methods of any type.

Example: Getting the Text Contents of a Cell in a Custom Data Grid

To get the text contents of a cell of a custom data grid from the Infragistics library, you can use the following C# code in the AUT:

```
string cellText =
dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

The following C# code sample gets the text contents of the third cell in the first row:

```
string cellText = dataGrid.Rows[0].Cells[2];
```

Scripting the same example by using the `InvokeMethods` method generates a relatively complex script, because you have to pass five methods with their corresponding parameters to the `InvokeMethods` method:

```
' VB .NET code
Dim dataGrid = mainWindow.WPFControl("@automationId='Custom
Data Grid'")

' Get text contents of third cell in first row.
Dim rowIndex = 0
Dim column = 2

Dim methodNames = New List(Of String)()
methodNames.Add("Rows")                    ' Get the list of
rows from the grid.
methodNames.Add("get_Item")                ' Get a specific row
```

```
from the list of rows by using the indexer method.
methodNames.Add("Cells")                  ' Get the list of
cells from the the row.
methodNames.Add("get_Item")               ' Get a specific cell
from the list of cells by using the indexer method.
methodNames.Add("Text")                   ' Get the text of the
cell.

Dim parameters = New List(Of List(Of Object))()
parameters.Add(New List(Of Object)())                  '
Parameters for the Rows property.
parameters.Add(New List(Of Object) From {rowIndex})    '
Parameters for the get_Item method.
parameters.Add(New List(Of Object)())                  '
Parameters for the Cells property.
parameters.Add(New List(Of Object) From {columnIndex}) '
Parameters for the get_Item method.
parameters.Add(New List(Of Object)())                  '
Parameters for the Text property.

Dim cellText As String = dataGrid.InvokeMethods(methodNames,
parameters)
```

A better approach in such a case is to add code to the application under test and then to use the `InvokeMethods` method. For this example, add the `GetCellText` method to the AUT:

```
// C# code, if the AUT is implemented in C#.
public static string
GetCellText(Infragistics.Win.UltraWinGrid.UltraGrid dataGrid,
int rowIndex, int columnIndex) {
  return dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

```
' VB code, if the AUT is implemented in VB.
public static string
GetCellText(Infragistics.Win.UltraWinGrid.UltraGrid dataGrid,
int rowIndex, int columnIndex) {
  return dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

To get the text contents of the cell, dynamically invoke the `GetCellText` method from your test script:

```
'VB .NET code
Dim cellText As String = mainWindow.Invoke("GetCellText",
dataGrid, rowIndex, columnIndex)
```

For additional information, see *Adding Code to the Application Under Test to Test Custom Controls*.

Supported Methods and Properties

The following methods and properties can be called:

- Methods and properties that Silk Test Workbench supports for the control.
- All public methods and properties that the MSDN defines for the control.
- If the control is a custom control that is derived from a standard control, all methods and properties from the standard control can be called.

Supported Parameter Types

The following parameter types are supported:

- All built-in Silk Test Workbench types

Silk Test Workbench types includes primitive types (such as boolean, int, string), lists, and other types (such as Point and Rect).

- Enum types

  Enum parameters must be passed as string. The string must match the name of an enum value. For example, if the method expects a parameter of the .NET enum type `System.Windows.Visiblity` you can use the string values of `Visible`, `Hidden`, or `Collapsed`.

- .NET structs and objects

  .NET struct and object parameters must be passed as a list. The elements in the list must match one constructor for the .NET object in the test application. For example, if the method expects a parameter of the .NET type `System.Windows.Vector`, you can pass a list with two integers. This works because the `System.Windows.Vector` type has a constructor with two integer arguments.

- Other controls

  Control parameters can be passed or returned as TestObject.

Returned Values

The following values are returned for properties and methods that have a return value:

- The correct value for all built-in Silk Test Workbench types. These types are listed in the *Supported Parameter Types* section.
- All methods that have no return value return `null` in C# or `Nothing` in VB.

Windows Presentation Foundation (WPF) Support

Silk Test Workbench provides built-in support for testing Windows Presentation Foundation (WPF) applications. Silk Test Workbench supports standalone WPF applications and can record and play back controls embedded in .NET version 3.5 or later.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Supported Controls

For a complete list of the controls available for WPF testing, see *WPF Class Reference*.

All supported WPF classes for Silk Test Workbench WPF support start with the prefix *WPF*, such as `WPFWindow` and `WPFListBox`.

Supported methods and properties for WPF controls depend on the actual implementation and runtime state. The methods and properties may differ from the list that is defined for the corresponding class. To determine the methods and properties that are supported in a specific situation, use the following code:

- `GetPropertyList()`
- `GetDynamicMethodList()`

For additional information abut WPF, refer to *MSDN*.

WPF Class Reference

When you configure a WPF application, Silk Test Workbench automatically provides built-in support for testing standard WPF controls.

Attributes for Windows Presentation Foundation (WPF) Applications

Supported attributes for WPF applications include:

- *automationId*
- *caption*

- *className*
- *name*
- All dynamic locator attributes.

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

Object Recognition

To identify components within WPF scripts, you can specify the *automationId*, *caption*, *className*, or *name*. The name that is given to an element in the application is used as the *automationId* attribute for the locator if available. As a result, most objects can be uniquely identified using only this attribute. For example, a locator with an *automationId* might look like: `//WPFButton[@automationId='okButton']"`.

If you define an *automationId* and any other attribute, only the *automationId* is used during replay. If there is no *automationId* defined, the *name* is used to resolve the component. If neither a *name* nor an *automationId* are defined, the *caption* value is used. If no caption is defined, the *className* is used. We recommend using the *automationId* because it is the most useful property.

Attribute Type	Description	Example
automationId	An ID that was provided by the developer of the test application.	`//WPFButton[@automationId='okButton']"`
name	The name of a control. The Visual Studio designer automatically assigns a name to every control that is created with the designer. The application developer uses this name to identify the control in the application code.	`//WPFButton[@name='okButton']"`
caption	The text that the control displays. When testing a localized application in multiple languages, use the automationId or name attribute instead of the caption.	`//WPFButton[@automationId='Ok']"`
className	The simple .NET class name (without namespace) of the WPF control. Using the class name attribute can help to identify a custom control that is derived from a standard WPF control that Silk Test Workbench recognizes.	`//WPFButton[@className='MyCustomButton']"`

During recording, Silk Test Workbench creates a locator for a WPF control by using the *automationId*, *name*, *caption*, or *className* attributes in the order that they are listed in the preceding table. For example,

if a control has a *automationId* and a *name*, Silk Test Workbench uses the *automationId* when creating the locator.

The following example shows how an application developer can define a *name* and an *automationId* for a WPF button in the XAML code of the application:

```
<Button Name="okButton" AutomationProperties.AutomationId="okButton"
Click="okButton_Click">Ok</Button>
```

Custom Attributes for WPF Applications

WPF applications use the predefined automation property `AutomationProperties.AutomationId` to specify a stable identifier for the WPF control as follows:

```
<Window x:Class="Test.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
  <Grid>
    <Button AutomationProperties.AutomationId="AID_buttonA">The
Button</Button>
  </Grid>
</Window>
```

Silk Test Workbench automatically uses this property for identification in the locator. WPF application locators look like the following:

```
/WPFWindow[@caption='MainWindow']//WPFButton[@automationId='AID_buttonA']
```

Classes that Derive from the WPFItemsControl Class

Silk Test Workbench can interact with classes that derive from `WPFItemsControl`, such as `WPFListBox`, `WPFTreeView`, and `WPFMenu`, in two ways:

- Working with the control

  Most controls contain methods and properties for typical use cases. The items are identified by text or index.

  For example:

  ```
  listBox.Select("Banana")
  listBox.Select(2)

  tree.Expand("/Fruit/Banana")
  ```

- Working with individual items, such as `WPFListBoxItem`, `WPFTreeViewItem`, or `WPFMenuItem`

  For advanced use cases, use individual items. For example, use individual items for opening the context menu on a specific item in a list box, or clicking a certain position relative to an item.

  For example:

  ```
  Dim item as WPFListBoxItem = listBox.WPFListBoxItem("banana")
  item.OpenContextMenu()
  ```

Custom WPF Controls

Generally, Silk Test Workbench provides record and playback support for all standard WPF controls.

Silk Test Workbench handles custom controls based on the way the custom control is implemented. You can implement custom controls by using the following approaches:

- Deriving classes from `UserControl`

  This is a typical way to create compound controls. Silk Test Workbench recognizes these user controls as `WPFUserControl` and provides full support for the contained controls.

- Deriving classes from standard WPF controls, such as `ListBox`

  Silk Test Workbench treats these controls as an instance of the standard WPF control that they derive from. Record, playback, and recognition of children may not work if the user control behavior differs significantly from its base class implementation.
- Using standard controls that use templates to change their visual appearance

  Low-level replay might not work in certain cases. Switch to high-level playback mode in such cases. To change the playback mode, use the **Options** dialog box and change the **Playback mode** in the **Playback General** settings list.

Silk Test Workbench filters out certain controls that are typically not relevant for functional testing. For example, controls that are used for layout purposes are not included. However, if a custom control derives from an excluded class, specify the name of the related WPF class to expose the filtered controls during recording and playback.

Dynamically Invoking WPF Methods

Dynamic invoke enables you to directly call methods, retrieve properties, or set properties, on an actual instance of a control in the application under test. You can also call methods and properties that are not available in the Silk Test Workbench API for this control. Dynamic invoke is especially useful when you are working with custom controls, where the required functionality for interacting with the control is not exposed through the Silk Test Workbench API.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

Call dynamic methods on objects with the `Invoke` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Call multiple dynamic methods on objects with the `InvokeMethods` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.

For example, to call a method named `SetTitle`, which requires the title to be set as an input parameter of type string, on an actual instance of a control in the application under test, type the following:

```
control.Invoke("SetTitle", "my new title")
```

Note: Typically, most properties are read-only and cannot be set.

Note: Reflection is used in most technology domains to call methods and retrieve properties.

The Invoke Method

For a Windows Forms or a WPF control, you can use the `Invoke` method to call the following methods:

- Public methods that the MSDN defines for the control.
- Public static methods that the MSDN defines.
- User-defined public static methods of any type.

> **First Example for the Invoke Method**
>
> For an object of the Silk Test Workbench type `DataGrid`, you can call all methods that MSDN defines for the type `System.Windows.Forms.DataGrid`.

To call the method `IsExpanded` of the `System.Windows.Forms.DataGrid` class, use the following code:

```
//VB .NET code
Dim isExpanded As Boolean = dataGrid.Invoke("IsExpanded", 3)
```

Second Example for the Invoke Method

To invoke the static method `String.Compare(String s1, String s2)` inside the AUT, use the following code:

```
//VB .NET code
Dim result as Integer = (Integer)
mainWindow.Invoke("System.String.Compare", "a", "b")
```

Third Example for the Invoke Method

This example shows how you can dynamically invoke the user-generated method `GetContents`.

You can write code which you can use to interact with a control in the application under test (AUT), in this example an `UltraGrid`. Instead of creating complex dynamic invoke calls to retrieve the contents of the `UltraGrid`, you can generate a new method `GetContents` and then just dynamically invoke the new method.

In Visual Studio, the following code in the AUT defines the `GetContents` method as a method of the `UltraGridUtil` class:

```
//C# code, because this is code in the AUT
namespace UltraGridExtensions {
  public class UltraGridUtil {
    /// <summary>
    /// Retrieves the contents of an UltraGrid as nested list
    /// </summary>
    /// <param name="grid"></param>
    /// <returns></returns>
    public static List<List<string>>
GetContents(Infragistics.Win.UltraWinGrid.UltraGrid grid) {
      var result = new List<List<string>>();
      foreach (var row in grid.Rows) {
        var rowContent = new List<string>();
        foreach (var cell in row.Cells) {
          rowContent.Add(cell.Text);
        }
        result.Add(rowContent);
      }
      return result;
    }
  }
}
```

The code for the `UltraGridUtil` class needs to be added to the AUT. You can do this in the following ways:

- The application developer can compile the code for the class into the AUT. The assembly needs to be already loaded.
- You can create a new assembly that is loaded into the AUT during test execution.

To load the assembly, you can use the following code:

```
FormsWindow.LoadAssembly(String assemblyFileName)
```

You can load the assembly by using the full path, for example:

```
mainWindow.LoadAssembly("C:/temp/ultraGridExtensions.dll")
```

You can also find the location of the assembly by using the `Location` method:

```
//VB.NET code
Dim assemblyLocation =
GetType(UltraGridExtensions.UltraGridUtil).Assembly.Location
mainWindow.LoadAssembly(assemblyLocation)
```

When the code for the UltraGridUtil class is in the AUT, you can add the following code to your test script to invoke the `GetContents` method:

```
var contents = (IList)
mainWindow.Invoke("UltraGridExtensions.UltraGridUtil.GetContents
", ultraGrid);
```

The `mainWindow` object, on which the `Invoke` method is called, only identifies the AUT and can be replaced by any other object in the AUT.

The InvokeMethods Method

For a Windows Forms or a WPF control, you can use the `InvokeMethods` method to invoke a sequence of nested methods. You can call the following methods:

- Public methods that the MSDN defines for the control.
- Public static methods that the MSDN defines.
- User-defined public static methods of any type.

Example: Getting the Text Contents of a Cell in a Custom Data Grid

To get the text contents of a cell of a custom data grid from the Infragistics library, you can use the following C# code in the AUT:

```
string cellText =
dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

The following C# code sample gets the text contents of the third cell in the first row:

```
string cellText = dataGrid.Rows[0].Cells[2];
```

Scripting the same example by using the `InvokeMethods` method generates a relatively complex script, because you have to pass five methods with their corresponding parameters to the `InvokeMethods` method:

```
' VB .NET code
Dim dataGrid = mainWindow.WPFControl("@automationId='Custom
Data Grid'")

' Get text contents of third cell in first row.
Dim rowIndex = 0
Dim column = 2

Dim methodNames = New List(Of String)()
methodNames.Add("Rows")                    ' Get the list of
rows from the grid.
methodNames.Add("get_Item")                ' Get a specific row
from the list of rows by using the indexer method.
methodNames.Add("Cells")                   ' Get the list of
cells from the the row.
methodNames.Add("get_Item")                ' Get a specific cell
from the list of cells by using the indexer method.
methodNames.Add("Text")                    ' Get the text of the
cell.
```

```
Dim parameters = New List(Of List(Of Object))()
parameters.Add(New List(Of Object)())                    '
Parameters for the Rows property.
parameters.Add(New List(Of Object) From {rowIndex})    '
Parameters for the get_Item method.
parameters.Add(New List(Of Object)())                    '
Parameters for the Cells property.
parameters.Add(New List(Of Object) From {columnIndex}) '
Parameters for the get_Item method.
parameters.Add(New List(Of Object)())                    '
Parameters for the Text property.

Dim cellText As String = dataGrid.InvokeMethods(methodNames,
parameters)
```

A better approach in such a case is to add code to the application under test and then to use the `InvokeMethods` method. For this example, add the `GetCellText` method to the AUT:

```
// C# code, if the AUT is implemented in C#.
public static string
GetCellText(Infragistics.Win.UltraWinGrid.UltraGrid dataGrid,
int rowIndex, int columnIndex) {
  return dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

```
' VB code, if the AUT is implemented in VB.
public static string
GetCellText(Infragistics.Win.UltraWinGrid.UltraGrid dataGrid,
int rowIndex, int columnIndex) {
  return dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

To get the text contents of the cell, dynamically invoke the `GetCellText` method from your test script:

```
'VB .NET code
Dim cellText As String = mainWindow.Invoke("GetCellText",
dataGrid, rowIndex, columnIndex)
```

For additional information, see *Adding Code to the Application Under Test to Test Custom Controls*.

Supported Methods and Properties

The following methods and properties can be called:

- Methods and properties that Silk Test Workbench supports for the control.
- All public methods and properties that the MSDN defines for the control.
- If the control is a custom control that is derived from a standard control, all methods and properties from the standard control can be called.

Supported Parameter Types

The following parameter types are supported:

- All built-in Silk Test Workbench types

  Silk Test Workbench types includes primitive types (such as boolean, int, string), lists, and other types (such as Point and Rect).
- Enum types

  Enum parameters must be passed as string. The string must match the name of an enum value. For example, if the method expects a parameter of the .NET enum type `System.Windows.Visiblity` you can use the string values of `Visible`, `Hidden`, or `Collapsed`.

- .NET structs and objects

  .NET struct and object parameters must be passed as a list. The elements in the list must match one constructor for the .NET object in the test application. For example, if the method expects a parameter of the .NET type `System.Windows.Vector`, you can pass a list with two integers. This works because the `System.Windows.Vector` type has a constructor with two integer arguments.
- WPF controls

  WPF control parameters can be passed as `TestObject`.

Returned Values

The following values are returned for properties and methods that have a return value:

- The correct value for all built-in Silk Test Workbench types. These types are listed in the *Supported Parameter Types* section.
- All methods that have no return value return `null` in C# or `Nothing` in VB.
- A string for all other types

  Call `ToString` on returned .NET objects to retrieve the string representation

Example

For example, when an application developer creates a custom calculator control that offers the following methods and the following property:

```
public void Reset()
public int Add(int number1, int number2)
public System.Windows.Vector StrechVector(System.Windows.Vector
vector, double
factor)
public String Description { get;}
```

The tester can call the methods directly from his test. For example:

```
customControl.Invoke("Reset")
Dim sum as Integer = customControl.Invoke("Add", 1, 2)
' the vector can be passed as list of integer
Dim vector = New List(Of Integer)
vector.Add(3)
vector.Add(4)
' returns "6;8" because this is the string representation of
the .NET object
Dim strechedVector As String =
customControl.Invoke("StrechVector", vector, 2.0)
Dim description As String =
customControl.GetProperty("Description")
```

Setting WPF Classes to Expose During Recording and Playback

Silk Test Workbench filters out certain controls that are typically not relevant for functional testing. For example, controls that are used for layout purposes are not included. However, if a custom control derives from an excluded class, specify the name of the related WPF class to expose the filtered controls during recording and playback.

Specify the names of any WPF classes that you want to expose during recording and playback. For example, if a custom class called *MyGrid* derives from the WPF `Grid` class, the objects of the *MyGrid* custom class are not available for recording and playback. `Grid` objects are not available for recording and playback because the `Grid` class is not relevant for functional testing since it exists only for layout purposes. As a result, `Grid` objects are not exposed by default. In order to use custom classes that are based on classes that are not relevant to functional testing, add the custom class, in this case *MyGrid*, to

the **OPT_WPF_CUSTOM_CLASSES** option. Then you can record, playback, find, verify properties, and perform any other supported actions for the specified classes.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree.
3. Click **WPF**.
4. In the **Custom WPF class names** grid, type the name of the class that you want to expose during recording and playback.

   Separate class names with a comma.
5. Click **OK**.

Setting Pre-Fill During Recording and Replaying

Defines whether items in a `WPFItemsControl`, like `WPFComboBox` or `WPFListBox`, are pre-filled during recording and playback. WPF itself lazily loads items for certain controls, so these items are not available for Silk Test Workbench if they are not scrolled into view. Turn pre-filling on, which is the default setting, to additionally access items that are not accessible without scrolling them into view. However, some applications have problems when the items are pre-filled by Silk Test Workbench in the background, and these applications can therefore crash. In this case turn pre-filling off.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree.
3. Click **WPF**.
4. Set **Pre-fill items** to **Yes**.
5. Click **OK**.

Silverlight Application Support

Microsoft Silverlight (Silverlight) is an application framework for writing and running rich internet applications, with features and purposes similar to those of Adobe Flash. The run-time environment for Silverlight is available as a plug-in for most web browsers.

Silk Test Workbench provides built-in support for testing Silverlight applications. Silk Test Workbench supports Silverlight applications that run in a browser as well as out-of-browser and can record and play back controls in .NET version 3.5 or later.

The following applications, that are based on Silverlight, are supported:

- Silverlight applications that run in Internet Explorer.
- Out-of-Browser Silverlight applications.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Supported Controls

Silk Test Workbench includes record and replay support for Silverlight controls.

For a complete list of the controls available for Silverlight testing, see the *Silverlight Class Reference*.

Note: With Silk Test 14.0 or later, Silk Test Workbench recognizes only Silverlight controls that are available for interaction and visible on the screen. This change might change the behavior of tests that were recorded with a Silk Test version prior to Silk Test 14.0. To run such tests with Silk Test 14.0 or later, remove all invisible or not yet available Silverlight controls from the tests.

Prerequisites

The support for testing Silverlight applications in Microsoft Windows XP requires the installation of Service Pack 3 and the Update for Windows XP with the Microsoft User Interface Automation that is provided in

Windows 7. You can download the update from *http://www.microsoft.com/download/en/details.aspx?id=13821*.

Note: The Microsoft User Interface Automation needs to be installed for the Silverlight support. If you are using a Windows operating system and the Silverlight support does not work, you can install the update with the Microsoft User Interface Automation, which is appropriate for your operating system, from *http://support.microsoft.com/kb/971513*.

Silverlight Class Reference

When you configure a Silverlight application, Silk Test Workbench automatically provides built-in support for testing standard Silverlight controls.

Locator Attributes for Identifying Silverlight Controls

Supported locator attributes for Silverlight controls include:

- *automationId*
- *caption*
- *className*
- *name*
- All dynamic locator attributes

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

To identify components within Silverlight scripts, you can specify the *automationId*, *caption*, *className*, *name* or any dynamic locator attribute. The *automationId* can be set by the application developer. For example, a locator with an *automationId* might look like `//SLButton[@automationId="okButton"]`.

We recommend using the *automationId* because it is typically the most useful and stable attribute.

Attribute Type	Description	Example
automationId	An identifier that is provided by the developer of the application under test. The Visual Studio designer automatically assigns an *automationId* to every control that is created with the designer. The application developer uses this ID to identify the control in the application code.	`//SLButton[@automationId="okButton"]`
caption	The text that the control displays. When testing a localized application in multiple languages, use the *automationId* or *name* attribute instead of the *caption*.	`//SLButton[@caption="Ok"]`
className	The simple .NET class name (without namespace) of the Silverlight control. Using the *className* attribute can help to identify a custom control that is derived from a standard Silverlight control that Silk Test Workbench recognizes.	`//SLButton[@className='MyCustomButton']`
name	The name of a control. Can be provided by the developer of the application under test.	`//SLButton[@name="okButton"]`

Attention: The *name* attribute in XAML code maps to the locator attribute *automationId*, not to the locator attribute *name*.

During recording, Silk Test Workbench creates a locator for a Silverlight control by using the *automationId*, *name*, *caption*, or *className* attributes in the order that they are listed in the preceding table. For example, if a control has an *automationId* and a *name*, Silk Test Workbench uses the *automationId*, if it is unique, when creating the locator.

The following table shows how an application developer can define a Silverlight button with the text "Ok" in the XAML code of the application:

XAML Code for the Object	Locator to Find the Object from Silk Test
`<Button>Ok</Button>`	`//SLButton[@caption="Ok"]`
`<Button Name="okButton">Ok</Button>`	`//SLButton[@automationId="okButton"]`
`<Button AutomationProperties.AutomationId="okButton">Ok</Button>`	`//SLButton[@automationId="okButton"]`
`<Button AutomationProperties.Name="okButton">Ok</Button>`	`//SLButton[@name="okButton"]`

Dynamically Invoking Silverlight Methods

Dynamic invoke enables you to directly call methods, retrieve properties, or set properties, on an actual instance of a control in the application under test. You can also call methods and properties that are not available in the Silk Test Workbench API for this control. Dynamic invoke is especially useful when you are working with custom controls, where the required functionality for interacting with the control is not exposed through the Silk Test Workbench API.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

Call dynamic methods on objects with the `Invoke` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Call multiple dynamic methods on objects with the `InvokeMethods` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.

For example, to call a method named `SetTitle`, which requires the title to be set as an input parameter of type string, on an actual instance of a control in the application under test, type the following:

```
control.Invoke("SetTitle", "my new title")
```

Note: Typically, most properties are read-only and cannot be set.

Note: Reflection is used in most technology domains to call methods and retrieve properties.

Supported Parameter Types

The following parameter types are supported:

- All built-in Silk Test Workbench types.

  Silk Test Workbench types include primitive types, for example boolean, int, and string, lists, and other types, for example Point and Rect.
- Enum types.

  Enum parameters must be passed as string. The string must match the name of an enum value. For example, if the method expects a parameter of the .NET enum type `System.Windows.Visiblity` you can use the string values of `Visible`, `Hidden`, or `Collapsed`.
- .NET structs and objects.

  Pass .NET struct and object parameters as a list. The elements in the list must match one constructor for the .NET object in the test application. For example, if the method expects a parameter of the .NET

type `System.Windows.Vector`, you can pass a list with two integers. This works because the `System.Windows.Vector` type has a constructor with two integer arguments.

- Other controls.

  Control parameters can be passed as `TestObject`.

Supported Methods and Properties

The following methods and properties can be called:

- All public methods and properties that the MSDN defines for the `AutomationElement` class. For additional information, see *http://msdn.microsoft.com/en-us/library/system.windows.automation.automationelement.aspx*.
- All methods and properties that MSUIA exposes. The available methods and properties are grouped in "patterns". Pattern is a MSUIA specific term. Every control implements certain patterns. For an overview of patterns in general and all available patterns see *http://msdn.microsoft.com/en-us/library/ms752362.aspx*. A custom control developer can provide testing support for the custom control by implementing a set of MSUIA patterns.

Returned Values

The following values are returned for properties and methods that have a return value:

- The correct value for all built-in Silk Test Workbench types.
- All methods that have no return value return `null` in C# or `Nothing` in VB.
- A string for all other types.

  To retrieve this string representation, call the `ToString` method on returned .NET objects in the application under test.

Example

A `TabItem` in Silverlight, which is an item in a `TabControl`.

```
tabItem.Invoke("SelectionItemPattern.Select")
mySilverligtObject.GetProperty("IsPassword")
```

Scrolling in Silverlight

Silk Test Workbench provides two different sets of scrolling-related methods and properties, depending on the Silverlight control.

- The first type of controls includes controls that can scroll by themselves and therefore do not expose the scrollbars explicitly as children. For example combo boxes, panes, list boxes, tree controls, data grids, auto complete boxes, and others.
- The second type of controls includes controls that cannot scroll by themselves but expose scrollbars as children for scrolling. For example text fields.

This distinction in Silk Test Workbench exists because the controls in Silk Test Workbench implement scrolling in those two ways.

Controls that support scrolling

In this case, scrolling-related methods and property are available for the control that contains the scrollbars. Therefore, Silk Test Workbench does not expose scrollbar objects.

Examples

The following command scrolls a list box to the bottom:

```
listBox.SetVerticalScrollPercent(100)
```

The following command scrolls the list box down by one unit:

```
listBox.ScrollVertical(ScrollAmount.SmallIncrement)
```

Controls that do not support scrolling

In this case the scrollbars are exposed. No scrolling-related methods and properties are available for the control itself. The horizontal and vertical scrollbar objects enable you to scroll in the control by specifying the increment or decrement, or the final position, as a parameter in the corresponding API functions. The increment or decrement can take the values of the ScrollAmount enumeration. For additional information, refer to the Silverlight documentation. The final position is related to the position of the object, which is defined by the application designer.

Examples

The following command scrolls a vertical scrollbar within a text box to position 15:

```
textBox.SLVerticalScrollBar().ScrollToPosition(15)
```

The following command scrolls a vertical scrollbar within a text box to the bottom:

```
textBox.SLVerticalScrollBar().ScrollToMaximum()
```

Troubleshooting when Testing Silverlight Applications

Silk Test Workbench cannot see inside the Silverlight application and no green rectangles are drawn during recording

The following reasons may cause Silk Test Workbench to be unable to see inside the Silverlight application:

Reason	Solution
You use a Silverlight version prior to 3.	Use Silverlight 3 (Silverlight Runtime 4) or Silverlight 4 (Silverlight Runtime 4).

Your Silverlight application is running in windowless mode.	Silk Test Workbench does not support Silverlight applications that run in windowless mode. To test such an application, you need to change the Web site where your Silverlight application is running. Therefore you need to set the `windowless` parameter in the object tag of the HTML or ASPX file, in which the Silverlight application is hosted, to `false`.
	The following sample code sets the `windowless` parameter to `false`:
	`<object ...>`
	`  <param name="windowless"`
	`value="false"/>`
	`  ...`
	`</object>`

Visual COBOL Support

Silk Test Workbench supports recording and replaying tests against Visual COBOL applications. You can also use the Silk Test Workbench APIs from .NET COBOL to script automated tests for the following Visual COBOL applications:

- Dialog system applications.

- CGI applications.
- Cobol Win32 applications.
- .NET COBOL - WPF and Windows Forms applications.
- COBOL JVM - Swing applications.
- Non-COBOL front-end applications calling back-end COBOL.

Note: For some controls, Silk Test Workbench provides only low-level recording support.

For information on the supported versions of Visual COBOL, refer to the *Release Notes*.

Rumba Support

Rumba is the world's premier Windows desktop terminal emulation solution. Silk Test provides built-in support for recording and replaying Rumba.

When testing with Rumba, please consider the following:

- The Rumba version must be compatible to the Silk Test version. Versions of Rumba prior to version 8.1 are not supported.
- All controls that surround the green screen in Rumba are using basic WPF functionality (or Win32).
- The supported Rumba desktop types are:
  - Mainframe Display
  - AS400 Display
  - Unix Display

For a complete list of the record and replay controls available for Rumba testing, see the *Rumba Class Reference*.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Rumba Class Reference

When you configure a Rumba application, Silk Test Workbench automatically provides built-in support for testing standard Rumba controls.

Enabling and Disabling Rumba

Rumba is the world's premier Windows desktop terminal emulation solution. Rumba provides connectivity solutions to mainframes, mid-range, UNIX, Linux, and HP servers.

Enabling Support

Before you can record and replay Rumba scripts, you need to enable support:

1. Install Rumba desktop client software version 8.1 or later.
2. Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Administration** > **Rumba plugin** > **Enable Silk Test Rumba plugin** or (in Microsoft Windows 10) **Start** > **Silk** > **Enable Silk Test Rumba plugin**.

Disabling Support

Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Administration** > **Rumba plugin** > **Disable Silk Test Rumba plugin** or (in Microsoft Windows 10) **Start** > **Silk** > **Disable Silk Test Rumba plugin**.

Locator Attributes for Identifying Rumba Controls

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests. Supported attributes include:

caption	The text that the control displays.
priorlabel	Since input fields on a form normally have a label explaining the purpose of the input, the intention of **priorlabel** is to identify the text input field, **RumbaTextField**, by the text of its adjacent label field, **RumbaLabel**. If no preceding label is found in the same line of the text field, or if the label at the right side is closer to the text field than the left one, a label on the right side of the text field is used.
StartRow	This attribute is not recorded, but you can manually add it to the locator. Use **StartRow** to identify the text input field, **RumbaTextField**, that starts at this row.
StartColumn	This attribute is not recorded, but you can manually add it to the locator. Use **StartColumn** to identify the text input field, **RumbaTextField**, that starts at this column.
All dynamic locator attributes.	For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Using Screen Verifications with Rumba

To automatically insert screen verifications in Rumba, turn on the following option in the **Options** dialog box: **Record** > **General** > **Record Screen Verifications**.

To manually insert screen verifications:

1. In your test, click the **Create Verification Type Logic** button to open the **Test Logic Designer - Verification**.
2. Click **Next**.
3. Select **The Contents of a Screen**.

   Any excluded objects as identified in **Tools** > **Options** > **Record** > **Rumba** > **Excluded Objects** will be used. You can customize these further in the **Properties** window of the test after you finish performing this procedure.
4. Click **Next**.
5. Click the **Identify** button.
6. Select the control on the Rumba Screen that you want to identify. The whole screen will be captured.
7. Click **Next**.
8. Click **Finish**.

Testing a Unix Display

Unix displays in Rumba are completely text-based, and provide no UI controls except the main **RUMBA screen** control. To replay a test on a Unix display, you can use the `SendKeys` method to send keys to the Unix display. Silk Test Workbench does not support recording on a Unix display.

SAP Support

Silk Test Workbench provides built-in support for testing SAP client/server applications based on the Windows-based GUI module.

Note: You can only test SAP applications with Silk Test Workbench if you have a Premium license for Silk Test Workbench. For additional information on the licensing modes, see *Licensing Information*.

Note: If you use SAP NetWeaver with Internet Explorer or Firefox, Silk Test Workbench tests the application using the xBrowser technology domain.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Supported Controls

For a complete list of the record and replay controls available for SAP testing, see the *SAP Class Reference*.

For a list of supported attributes, see *Attributes for SAP Applications*.

SAP Class Reference

When you configure a SAP application, Silk Test Workbench automatically provides built-in support for testing standard SAP controls.

Note: To enable Silk Test Workbench to recognize SAP controls, enable SAP GUI Scripting on the server. For information on enabling SAP GUI scripting, see *Introduction to SAP Gui Scripting*. If SAP GUI Scripting is not enabled, Silk Test Workbench recognizes all controls as Win32 and custom controls.

The classes included in the SAP class reference, along with all included properties and methods, are part of the SAP Automation module that is directly accessible through Silk Test Workbench.

Note: The interface, including the underlying algorithms and the behavior of the interface, is not under the control of Silk Test Workbench. Visual tests leverage properties and methods of the listed SAP classes only as far as simple data types are involved. For example, you cannot call the `GetCell` method of the `SAPTable` class from a visual test, but you can call the method from a VB .NET script.

Attributes for SAP Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for SAP include:

- automationId
- caption

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Dynamically Invoking SAP Methods

Dynamic invoke enables you to directly call methods, retrieve properties, or set properties, on an actual instance of a control in the application under test. You can also call methods and properties that are not available in the Silk Test Workbench API for this control. Dynamic invoke is especially useful when you are

working with custom controls, where the required functionality for interacting with the control is not exposed through the Silk Test Workbench API.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

Call dynamic methods on objects with the `Invoke` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Call multiple dynamic methods on objects with the `InvokeMethods` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.

For example, to call a method named `SetTitle`, which requires the title to be set as an input parameter of type string, on an actual instance of a control in the application under test, type the following:

```
control.Invoke("SetTitle", "my new title")
```

Note: Typically, most properties are read-only and cannot be set.

Note: Reflection is used in most technology domains to call methods and retrieve properties.

Supported Methods and Properties

The following methods and properties can be called:

- Methods and properties that Silk Test Workbench supports for the control.
- All public methods that the SAP automation interface defines
- If the control is a custom control that is derived from a standard control, all methods and properties from the standard control can be called.

Supported Parameter Types

The following parameter types are supported:

- All built-in Silk Test Workbench types

  Silk Test Workbench types includes primitive types (such as boolean, int, string), lists, and other types (such as Point and Rect).
- UI controls

  UI controls can be passed or returned as `TestObject`.

Returned Values

The following values are returned for properties and methods that have a return value:

- The correct value for all built-in Silk Test Workbench types. These types are listed in the *Supported Parameter Types* section.
- All methods that have no return value return `null` in C# or `Nothing` in VB.

Example

The following VB .NET script example shows how you can dynamically invoke SAP methods.

```
Dim _desktop As Desktop = Agent.Desktop
Dim wnd As SapWindow = _desktop.SapWindow("wnd 0")
Dim result As Object
```

```
result = wnd.Invoke("IsVKeyAllowed", 8) ' boolean return value
Console.WriteLine(
  "invoke result='" & result & "' type=" &
result.GetType().ToString())
wnd.Invoke("ShowMessageBox", "A Title", "Some text...", 1, 1)
```

Dynamically Invoking Methods on SAP Controls

When Silk Test Workbench cannot record actions against an SAP control, you can record the actions with the recorder that is available in SAP and then dynamically invoke the recorded methods in a Silk Test Workbench script. By doing so, you can replay actions against SAP controls that you cannot record.

1. To record the actions that you want to perform against the control, use the **SAP GUI Scripting** tool that is available in SAP.

   For additional information on the **SAP GUI Scripting** tool, refer to the SAP documentation.
2. Open the recorded actions from the location to which the **SAP GUI Scripting** tool has saved them and see what methods were recorded.
3. In Silk Test Workbench, dynamically invoke the recorded methods from your script.

Examples

For example, if you want to replay pressing a special control in the SAP UI, which is labeled *Test* and which is a combination of a button and a list box, and selecting the sub-menu *subsub2* of the control, you can record the action with the recorder that is available in SAP. The resulting code will look like the following:

```
session.findById("wnd[0]/usr/cntlCONTAINER/shellcont/
shell").pressContextButton "TEST"
session.findById("wnd[0]/usr/cntlCONTAINER/shellcont/
shell").selectContextMenuItem "subsub2"
```

Now you can use the following code to dynamically invoke the methods `pressContextButton` and `selectContextMenuItem` in your script in Silk Test Workbench:

```
.SapToolbarControl("shell
ToolbarControl").Invoke("pressContextButton", "TEST")
.SapToolbarControl("shell
ToolbarControl").Invoke("selectContextMenuItem", "subsub2")
```

Replaying this code will press the control in the SAP UI and select the sub-menu.

Configuring Automation Security Settings for SAP

Before you launch an SAP application, you must configure the security warning settings. Otherwise, a security warning, `A script is trying to attach to the GUI`, displays each time a test plays back an SAP application.

1. In **Windows Control Panel**, choose **SAP Configuration**. The **SAP Configuration** dialog box opens.
2. In the **Design Selection** tab, uncheck the **Notify When a Script Attaches to a Running SAP GUI**.

Working with SAP eCATT

Silk Test Workbench integrates with the *extended Computer Aided Test Tool (eCATT)* from SAP to provide SAP users with full testing support for products based on HTML technology. This gives eCATT test script developers a central point of control for all HTML product based testing. Silk Test Workbench is capable of interacting and saving data to either an ANSI or Unicode database.

When using Silk Test Workbench within the eCATT user interface, Silk Test Workbench assets are retrieved and stored together with the eCATT tests in the SAP/R3 database. Testers have a central overview of their results: eCATT and Silk Test Workbench test results are merged and can be shown in the same view in eCATT.

Developers can create truly integrated test scenarios by passing parameters between eCATT and Silk Test Workbench. The integration also allows testers to use data from SAP test data containers in Silk Test Workbench. Best of all, testers with access to the same SAP system can share their tests.

In Silk Test Workbench, pass parameters between eCATT and Silk Test Workbench using script arguments.

Two communication types are used for the interface between Silk Test Workbench and eCATT. The communication types are COM and RFC. The Silk Test Workbench GUI in eCATT is done through COM. The RFC communication is used by Silk Test Workbench to save and load data to and from the SAP/R3 database. Data includes one or more Silk Test Workbench projects and all related Silk Test Workbench assets which are stored as large objects in the SAP/R3 database. These large objects are known as *BLOBs*.

Integrating Silk Test Workbench and SAP eCATT

The integration of Silk Test Workbench and SAP eCATT lets you manage the testing process from start to finish. Once you have integrated Silk Test Workbench and SAP eCATT, you can use Silk Test Workbench and eCATT in the following two ways:

- Start Silk Test Workbench from within eCATT to record, playback, debug, and edit test scripts. You can also save assets in a BLOB, load a BLOB from the SAP/R3 database, and manage test script arguments. This is the recommended way.
- Start the standalone version of Silk Test Workbench separately from eCATT. You can still transfer data in a BLOB between Silk Test Workbench and eCATT.

Setting Up the Integration with eCATT

To configure the integration between Silk Test Workbench and eCATT, perform the following steps:

1. Login to your SAP server as an administrator to make permanent changes.
2. Enable scripting on the application server. For additional information, see the topic *Enabling Scripting on the Application Server* in the SAP Help.
3. On your SAP server, register Silk Test Workbench as an external tool for eCATT. For additional information, see the topic *Using an External Tool with eCATT* in the SAP Help.
4. For each client, enable scripting on the front-end of the client.

   Note: Clear the **Notify when a script attaches to a running GUI** check box to prevent a confirmation dialog box from displaying each time Silk Test Workbench records against an SAP application.

   For additional information, see the topic *Enabling Scripting at the Front End* in the SAP Help.
5. On your SAP server, create a new user account.
6. On the client machine on which you will be using Silk Test Workbench in combination with eCATT, install both Silk Test Workbench and the SAPGUI client.

Registering Silk Test Workbench in eCATT

To integrate Silk Test Workbench and eCATT, Silk Test Workbench must be registered in the *ECCUST_ET* SAP table. The `SET_EXTERNAL_TOOL` method creates the necessary entries in the *ECCUST_ET* table. You can call the `SET_EXTERNAL_TOOL` function module using the SE37 transaction:

1. On the first screen, enter the function module name `SET_EXTERNAL_TOOL`.

2. From the **Function Module** menu, select **Test** > **Single Test**.
3. In the following window, enter the following parameter values:

Parameter	Value
TOOL_NAME	Silk Test Workbench
PROG_ID	SilkTest.STWEcattControl
TOOL_DESC	Silk Test Workbench for eCATT
TOOL_DATABASE	STW_EDIT
TOOL_RUN_DB	STW_EXEC
TOOL_NO_PWD	<blank>
TOOL_NO_DB	<blank>

4. Press the **Execute** button (F8).

Creating a Specific User Account

To take advantage of eCATT integration using external tools, your system administrator has to generate a standard user in your system. This has to be done once for each system by executing the ECATT_GENERATE_ET_USER program in SE38.

To activate the newly created user role after running the report, perform the following steps:

1. In the transaction PFCG, enter the role SAP_ECET, and select **Change**. You can ignore the subsequent warning that appears.
2. Switch to the **Authorizations** tab and select **Change Authorization Data**.
3. Place the cursor over the top node of the **SAP_ECET** tree and select **Authorizations** > **Generate**.
4. Click **Back** to return to the role maintenance screen.
5. Click **Save**.

Configuring the Standalone Version of Silk Test Workbench

The topics in this section describe additional configurations that you have to make when you want to transfer data in a BLOB between Silk Test Workbench and eCATT by starting the standalone version of Silk Test Workbench separately from eCATT. If you are starting Silk Test Workbench from within eCATT, you do not have to make these configurations.

Setting SAP/R3 Login Parameters

The SAP/R3 login parameters are used to log on to the SAP/R3 database. You can use Silk Test Workbench to log on to the SAP/R3 database to retrieve and transfer BLOBs between Silk Test Workbench and eCATT.

1. Perform one of the following:
   a) Click the **Change Login Parameters** button from the **Save BLOB** dialog box.
   b) Click **Tools** > **SAP eCATT** > **SAP/R3 Login Parameters** from the Silk Test Workbench menu.

   The **SAP/R3 Login** dialog box appears. If you are connected to eCATT, established logon parameters appear. You only need to re-enter your password to log on. If you are already connected, proceed with step 3.

   If you are currently disconnected from eCATT, you must perform a one-time log on to the SAP eCATT database. SAP keeps logon information, so once logged in, only a password is required to reconnect for data transfer.
2. Type your three-digit SAP Specific Client session number in the **Client** field. Refer to SAP SDK documentation for more information about the client number.

3. Type your SAP/R3 database logon User ID in the **Username** field.
4. Type your SAP/R3 password in the **Password** field.
5. Type a valid SAP language code in the **Language** field.
6. Click **OK** to connect, or click **Cancel** to exit without connecting.

Setting or Changing RFC Connection Parameters

The RFC interface is used to transport BLOBs, argument containers, and execution information between Silk Test Workbench and the SAP/R3 database. To save and load BLOBs between Silk Test Workbench and SAP eCATT, connection information is required. Use the following procedure to change connection parameters for the RFC interface.

1. If working with the eCATT interface, switch to the active session of Silk Test Workbench.
2. Perform one of the following:
   a) Click the **Change RFC Parameters** button from the **Save BLOB** dialog box.
   b) Click **Tools** > **SAP eCATT** > **RFC Connection Parameters** from Silk Test Workbench menu.

   The **RFC Connection Parameters** dialog box appears.
3. Perform one of the following:

   If you have already established an RFC connection for your current session, the parameters appear in the **RFC Connection Parameters** dialog box. If these parameters do not require any changes, proceed to step 4.

   If you have not established an RFC connection for your current session, or changes to the RFC connection parameters are required, enter the required connection information. This includes the following:

Use SAP Logon ID	If you have already created an **SAP Logon ID** with SAP GUI, check this box and then select a logon. If there are no entries in SAP GUI, this check box and the **SAP Logon ID** are disabled.
SAP Logon ID	Select a logon from the list.
AS Host	This is the name of the SAP Application Server host machine used for the RFC connection. This can contain an SAP routing string that precedes the host name.
RFC Type	The server type used for the Silk Test Workbench RFC connection to SAP eCATT. For an R3 system, the RFC types is 3.
System nr.	The SAP system number for the RFC connection.

4. If your SAP server is on a different network than the SAP GUI client and Silk Test Workbench, you need to manually type the SAP router string into the eCATT ABAB code. Perform the following steps:
   a) Start transaction se80.
   b) Edit `Class/Interface cl_apl_ecatt_extprog`.
   c) Go to method `Launch` and add the following code: `ashost = '<ROUTER-STRING+IP>'`. Where `<ROUTER-STRING+IP>` is the SAP Router string and the `IP` is the address of the SAP server.
5. Click **OK** to apply your changes and close the dialog box. If saving a BLOB and a SAP/R3 login has been made between Silk Test Workbench and eCATT, the BLOB is transferred to the SAP/R3 database. Once the BLOB has transferred, you can work with its test script in eCATT.

Recording and Testing eCATT Scripts when Silk Test Workbench is Started from eCATT

When you have started Silk Test Workbench from eCATT, you can define a test using eCATT and record it using Silk Test Workbench as the eCATT external testing tool. This provides a central point from which to manage test and lets you use scripting capabilities within Silk Test Workbench to record and test target applications. The test recorded in Silk Test Workbench can be either a .NET script or a visual test.

Defining the Test Script in eCATT

1. Start a test script in eCATT. To do this, select **Test Script** in the eCATT initial screen, type `S_ECATT` as the name and supply a version number.
2. Use eCATT to specify Silk Test Workbench as the external tool.
3. Click the **Apply Object** button on the toolbar. In SAP eCATT, the script editor appears.

Specifying Test Script Attributes in eCATT

1. Use eCATT to specify the test script attributes. For additional instructions, see the eCATT help at *http://help.sap.com*.
2. Click the **Script** button in the application toolbar. The **User Entries for External Tool** dialog box appears.
3. Enter the user name and password for the `ST_EDIT` database.

   This is required to use Silk Test Workbench as the external tool.
4. Enter the name of the external-tool project in which the test script will be created. This is the name of the project in Silk Test Workbench
5. Check the check box if you want to clear the `ST_EDIT` database.

Silk Test Workbench starts from eCATT, and the new test script created in eCATT appears in the **View code** window. You can now record and test the test script, or close the **View code** window and create and record a visual test to test the target application.

Recording and Testing the Script using Silk Test Workbench

1. Ensure that the target application is running and available.
2. In Silk Test Workbench, record the .NET script or create and record a visual test.
3. *Optional:* Create any arguments.

   For additional information, see *Supporting eCATT Script Arguments in Silk Test Workbench*.
4. Use Silk Test Workbench to playback the .NET script or visual test. You can also view results created for the playback.
5. Save the test and all associated Silk Test Workbench assets in a BLOB. Once saved, the BLOB data is transferred back to the SAP/R3 database for use in eCATT.

Creating an eCATT Script Using the Standalone Silk Test Workbench

You can create an eCATT script in Silk Test Workbench, record or write additional VB.NET code and test, then transmit the test script to the SAP/R3 database for use in eCATT. In this case, Silk Test Workbench runs in a standalone environment (not started from eCATT), until the test script, project and related assets are passed to eCATT in a BLOB.

1. Start Silk Test Workbench.
2. Create and record the test script.

   Note: You can also start with a test script created in eCATT that has been loaded from a BLOB into temporary storage in the Silk Test Workbench database.
3. Playback the test script.
4. View any results created for the test script playback.
5. Create any arguments needed for the test script in eCATT.
6. To transfer the test script, its project and related assets to eCATT, save it in a BLOB. Since Silk Test Workbench is running in standalone mode, the SAP/R3 Login dialog box appears.
7. Set the SAP/R3 database login parameters and connect to the SAP/R3 database for BLOB transfers.

Note: Silk Test Workbench requires a valid SAP/R3 login user name and password to connect to the SAP/R3 database. See *Setting SAP/R3 Login Parameters* for specific instructions.

8. An RFC connection must be established for the current session. Set the RFC connection parameters to transfer BLOB data to eCATT. Once the connections are made, eCATT can be used to work with the test script.

Working with BLOBs

A *Binary Large Object* (BLOB) is a container used to transfer test scripts or visual tests between SAP eCATT and Silk Test Workbench. Information in a BLOB also includes all related assets, except results, that belong to the .NET script or visual test, including other .NET scripts or visual tests.

In addition to the related test assets, a BLOB also contains all the option settings that belong to the test script(s) or visual test(s).

BLOBs are XML documents. The BLOB format is only used in Silk Test Workbench internally. The BLOB data itself cannot be interpreted by external systems.

Only the most recently saved version of a BLOB is stored in the SAP eCATT database. Previous versions are not preserved. Once a specific BLOB is opened for editing/debugging by one user, it cannot be accessed by other users until it is resubmitted to the database. When this happens, the previous version is overwritten.

A BLOB is created (saved) in Silk Test Workbench. When saved, a BLOB is transported from Silk Test Workbench to the SAP/R3 database through the RFC interface that exists between Silk Test Workbench and eCATT.

Entering BLOB Tracking Information

1. Make sure all options are set the way you want for the opened test script. In, particular, the Playback options in Silk Test Workbench impact playback in eCATT.
2. From the Silk Test Workbench menu, click one of the following:
   a) Click **Tools** > **SAP eCATT** > **Save BLOB** > **Save and Continue** to save the BLOB and continue working in Silk Test Workbench.
   b) Click **Tools** > **SAP eCATT** > **Save BLOB** > **Save and Goto eCATT** to save the BLOB and go to eCATT when the BLOB is transferred to the SAP/R3 database. Silk Test Workbench closes when the save is complete, and eCATT comes into focus.

   The **Save BLOB** dialog box appears. Information for the most recently loaded BLOB appears in the dialog box. You can type over the information to save a new BLOB.
3. Enter a unique name for the BLOB in the BLOB ID field.
4. Optionally enter a version number for the BLOB in the Version field. If a version isn't specified, Version 1 is used.

   Note: Version numbers are not verified or incremented. Only the current version of a BLOB is saved in the SAP/R3 database.
5. If you want to save diagnostic data locally in an XML file, select the Dump to File check box.
6. Continue entering information for the BLOB by entering the BLOB content.

Entering or Changing BLOB content

The **Projects to Export** list shows all projects in the Silk Test Workbench database that you have access rights to.

Note: The **Connection Info** section is only displayed when you are using the standalone version of Silk Test Workbench, and not when you have started Silk Test Workbench from eCATT.

1. Select the projects that contain the primary test script you want to include in the BLOB by clicking the appropriate check boxes. If your primary script contains calls to other test scripts, select those projects also. When you select projects, all defined Record and Playback setting groups appear in the **Record**

Settings and Playback Settings lists. These groups are set in Silk Test Workbench options for Record and Playback.

2. If you want a defined group of record options to be part of the BLOB, select that record setting group from the **Record Settings** list box.
3. If you want a defined group of playback options to be part of the BLOB, select that playback setting group from the **Playback Settings** list box.
4. Select the primary Silk Test Workbench project from the **Primary Project** box. The primary project is the Silk Test Workbench project to which the BLOBs primary script belongs.
5. The **Asset Type** box determines the type of test being saved to the BLOB. Select .NET script or visual test from the list.
6. The **Primary .NET script/visual test** box lists all available test scripts or visual tests available in selected projects, as well as any .NET scripts or visual tests from currently loaded BLOBs, which are temporarily stored in the Silk Test Workbench database. .NET scripts or visual tests appear in the list depending on the Asset type selection. Select the primary .NET script or visual test for this BLOB from the list.
7. Click **Save** to save the BLOB in the SAP/R3 database.

   Note:

   By default, the Silk Test Workbench DSN for STW_EDIT and STW_EXEC is uppercase and must match the case of the DSN installed on the SAP server. For example, if you receive the error "STW_Edit Not Available" when saving a BLOB, check the case of the DSN name. You may need to update the Silk Test Workbench DSN from "STW_EDIT" to "STW_Edit".

   If you change a default playback or record option in Silk Test Workbench, the settings become nameless. To save a BLOB, you are forced to give such nameless settings a name. Playback and record settings can be saved to a profile. For additional information, see the topics *Saving Playback Options* and *Saving and Reusing Record Options*.

   Silk Test Workbench displays the SAP/R3 Login dialog box, where you can set database login parameters and connect to the SAP/R3 database for BLOB transfers.

If you are already connected to the SAP/R3 database, Silk Test Workbench only requires your database password. If you are not already connected to the SAP/R3 database, Silk Test Workbench requires your SAP/R3 login user name and password for the initial connection. See the topic "*Setting SAP/R3 Login Parameters*" for specific instructions.

If an RFC connection has been established, the **AS Host and Client information** displays in the **Save BLOB** dialog box. If that information is not displaying, an RFC connection has not been established for the current session. You must then set the RFC connection parameters to transfer BLOB data to eCATT. See "*Setting or changing RFC Connection Parameters*."

Supporting eCATT Script Arguments in Silk Test Workbench

eCATT script arguments are an essential concept in the eCATT environment. They form the "interface" between eCATT and the Silk Test Workbench test. Arguments contain additional data needed by the current BLOB to run the BLOBs test script or visual test in eCATT. These arguments allow a test script or visual test to be used in the eCATT environment.

Arguments can be created in Silk Test Workbench or eCATT, and can be imported and exported between both environments.

The argument support in Silk Test Workbench consists of the following components:

Silk Test Workbench Argument Maintenance Feature	The Silk Test Workbench Argument Maintenance lets you create new arguments and edit or remove existing ones using the **eCATT Argument Container** dialog box. Use the **eCATT Argument Container** dialog box to define any arguments that can potentially be exported to or imported from an eCATT script through a BLOB. Once

defined, arguments can be passed between an eCATT script and a Silk Test Workbench .NET script or visual test using the Silk Test Workbench/eCATT COM Interface.

Since Silk Test Workbench can only handle one BLOB (main script with arguments and related assets) at a time, the **eCATT Argument Container** dialog box always shows the arguments for the current BLOB.

When Silk Test Workbench stores a BLOB to the SAP database, it reads values specified for the BLOB's arguments and transports them in an argument container to be stored with the BLOB. The Silk Test Workbench Argument Maintenance Feature only works with string data type script arguments.

Using eCATT script arguments

For Silk Test Workbench .NET scripts, the argument values are passed to and from the script using two methods in the Silk Test Workbench scripting language. The `Workbench.SetECATTArg()` method creates argument values to pass to a SAP eCATT script. The `Workbench.GetECATTArg()` method retrieves argument values passed from a SAP eCATT argument container.

For visual tests, you create a *Set eCATT* variable step to export a defined argument value to an eCATT script. You create a *Get Contents of eCATT* variable step to import a defined argument value from an eCATT script and save it to a local variable, so the value can be used in visual test.

Creating an eCATT Script Argument

1. While a BLOB is loaded, click **Tools** > **SAP eCATT** > **Argument Container**. The **eCATT Argument Container** dialog box displays. Any arguments defined for the currently loaded BLOB appear in the data table.
2. Click **Add**. The **Add Arguments** dialog box displays.
3. In the **Name** field, enter the name you want for the argument.
4. Select the direction of the argument transfer (**Export** or **Import**) from the list in the **Direction** field.
5. Enter the default value for the argument in the **Default Value** field. The default value is data used by a test script in eCATT. Consult eCATT documentation if you need more information about default values.
6. Enter the argument's actual value in the **Value** field. The value can be obtained from the test script, which can be viewed in eCATT.
7. Optionally type a description for the argument in the **Description** field. The description can be used to help identify a particular argument when the name of the argument is not descriptive for identification.
8. Click **Add** to add the argument to the argument container for the current BLOB. The argument appears as an entry in the container in the **eCATT Argument Container** dialog box.
9. Repeat steps 1 through 9 to add other arguments.
10. When you have completed creating all the necessary arguments for the currently loaded BLOB, click **Close** to return to the **eCATT Argument Container** dialog box.
11. Click **OK**.

Editing an eCATT Script Argument

Arguments for the most recently loaded BLOB's primary .NET script or visual test can be changed while it is still loaded. Use the **eCATT Argument Container** dialog box in Silk Test Workbench to edit eCATT script arguments for the currently loaded BLOB.

1. While the BLOB whose arguments you want to change is loaded, click **Tools** > **SAP eCATT** > **Argument Container**. The **eCATT Argument Container** dialog box appears. Any arguments defined for the currently loaded BLOB appear in the dialog box's data table.

2. Select the argument whose parameters you want to edit.
3. Click **Edit**. The **Edit Arguments** dialog box displays.
4. Change argument values as needed:

Name	The name of the argument
Direction	Determines whether the argument is an Import (from eCATT to Silk Test Workbench) or Export (from Silk Test Workbench to eCATT) argument
Default Value	A default value for the argument's data. The default value is data used by a test script in eCATT if a value is not specified for an eCATT script argument. Consult eCATT documentation if you need more information about default values.
Value	The argument's actual value. The value can be obtained from the script, which can be viewed in eCATT.
Description	The description can be used to help identify a particular argument when the name of the argument is not descriptive for identification.

5. Click **Edit**. The argument's parameters appear in the **eCATT Argument Container** dialog box.
6. Repeat steps 1 and 2 if you need to edit other arguments to the argument container.
7. When you have completed editing all the necessary arguments for the currently loaded BLOB, click **Close** to return to the **eCATT Argument Container** dialog box.
8. Click **OK**.

Removing an eCATT Argument from an Argument Container

Script arguments for the most recently loaded BLOB can be removed from its argument container at any time. Please note that Silk Test Workbench does not prompt you to confirm this action before removing the argument.

1. While a BLOB is loaded, click **Tools** > **SAP eCATT** > **Argument Container**. The **eCATT Argument Container** dialog box appears. Any arguments defined for the most recently loaded BLOB appear in the dialog box's data table. Each argument appears on a separate line.
2. Select the line with desired argument to remove and click **Remove**. The line with the argument is removed from the argument container.
3. Repeat step 2 for any other arguments you want removed.
4. Click **OK**.

Importing Arguments From an eCATT Script into a Visual Test

You can import eCATT script arguments from an argument container and use the argument values in any visual test. The contents of a passed argument are saved to a local variable in the visual test.

To import arguments from an eCATT script into a visual test:

1. Open the visual test into which you want to import the eCATT script arguments.
2. Determine which arguments you want to import from the eCATT script.
3. Use the **eCATT Argument Container** dialog box in Silk Test Workbench to create eCATT script import arguments that can be used in the visual test.
4. In the visual test, create local variables to store the imported arguments.

   The defined local variables display in the visual test's **<<Start>>** step. However, variables can be inserted into any test step.
5. Select a test step in the visual test, then click **Insert** > **SAP eCATT** > **Get Argument**.

   Silk Test Workbench creates a step immediately following the selected step. This newly created step is used to get the value of an eCATT argument and to store it in a variable so it can be used in the visual test. Properties for the step display in the **Properties** pane.

Tip: Place this test step at or near the beginning of the visual test to maintain optimal test step organization.

6. In the **Properties** pane for the test step, click the **Sort Category** icon to group properties by category.
7. Update the **Assignment category** properties.

   The **Assignment** properties for the step specify the eCATT argument to be imported and a local variable in which to store the argument value.

   a) In the **eCATT variable name** property, type the name of an eCATT script argument with a value to be used in the visual test.

      The specified eCATT script argument must match one of the argument names that you have defined in the **eCATT Argument Container** for the visual test.

   b) The `Local variable name` property lists all local variables that are defined for the visual test, including the variables that you have created to store the imported arguments. Select a local variable to store the value of the specified eCATT script argument from the list.

   ```
   Get contents of eCATT argument '[eCATT script argument name]' and put into
   local variable '[local variable name]'
   ```

8. Once the **Assignment category** properties are updated and the visual test is saved, the test step description is similar to the following:

   ```
   Get contents of eCATT argument '[eCATT script argument name]' and put into
   local variable '[local variable name]'
   ```

9. *Optional:* You can repeat the steps to import multiple eCATT script arguments into the visual test.

After eCATT script arguments are imported to a visual test and saved to local variables, the local variables can be used as input values in automation steps. For example, an eCATT script argument value for a login ID can be used as text input for an automation step that enters the value into a Login ID field.

Exporting Arguments From a Visual Test to an eCATT Script

You can pass eCATT script arguments that are used in a visual test to an eCATT script. The eCATT script arguments must be saved in a BLOB and must be defined as export arguments in the argument container for the visual test.

To export eCATT script arguments from a visual test to an eCATT script:

1. Open the visual test from which you want to export arguments to an eCATT script.
2. Determine which arguments you want to export to the eCATT script.

   Use the **eCATT Argument Container** dialog box in Silk Test Workbench to create eCATT script export arguments.
3. Select the **<<Start>>** step in the visual test, then click **Insert** > **SAP eCATT** > **Set Argument**.

   Silk Test Workbench creates a step to set the export value of an eCATT argument so it can be used in an eCATT script. The properties of the step display in the **Properties** pane.

   Tip: Place this test step at or near the beginning of the visual test to maintain optimal test step organization.
4. In the **Properties** pane for the step, click the **Sort Category** icon to group properties by category.
5. Update the Assignment category properties to define the name of the eCATT argument and the value to be assigned to it.

   a) In the **eCATT variable name** property, enter the name of an eCATT script argument that you want to export.

      The specified eCATT script argument must match an argument name defined in the **eCATT Argument container** for the visual test.

   b) In the **Value** property, type the for the argument, or right-click in the value area and assign a literal, variable, expression result, or ActiveData value.

Once the Assignment category properties are updated and the visual test is saved, the step is similar to the following:

```
Set eCATT argument '[eCATT script argument name]' to '[value]'
```

6. *Optional:* You can repeat the steps to export multiple arguments from the visual test to the eCATT script.

   Once the arguments are exported and the BLOB is saved, you can use the argument values in the eCATT script.

Importing Arguments From an eCATT Script Into a .NET Script

Retrieve eCATT script arguments from an argument container to use the argument values in any VB .NET script in Silk Test Workbench.

To import arguments from an eCATT script in a VB .NET script in Silk Test Workbench:

1. Open the VB .NET script to which the eCATT script arguments are to be imported.
2. Determine which arguments are to be imported from the eCATT script to the VB .NET script.

   Use the **eCATT Argument Container** dialog box in Silk Test Workbench to create eCATT script import arguments that can be used in the VB .NET script.
3. In the .NET script, create a local variable to store the imported argument.

   For example:

   ```
   Dim argName As String
   ```
4. Use the `GetECATTArg` function of the `Workbench` class to get the eCATT script argument.

   For example:

   ```
   ArgName = Workbench.GetECATTArg("NAME")
   ```
5. *Optional:* The `GetECATTArg` function returns a string. You can use a type cast in the .NET script to change the type to something else, for example a date.

After eCATT script arguments are imported to a .NET script, the arguments can be used as input values in automation steps. For example, an eCATT script argument value for a login ID can be used as text input for an automation step that enters the value into a Login ID field.

Exporting Arguments From a .NET Script to an eCATT Script

Pass eCATT script arguments used in a VB .NET script in Silk Test Workbench to an eCATT script to use the arguments in that script.

To export eCATT script arguments from a .NET script to an eCATT script:

1. Open the .NET script from which you want to export arguments to an eCATT script.
2. Determine which arguments are to be exported to the eCATT script.
3. Use the **eCATT Argument Container** dialog box in Silk Test Workbench to create eCATT script export arguments that can be used in the .NET script.
4. Use the `SetECATTArg` function of the `Workbench` class to export the eCATT script argument.

   For example:

   ```
   Workbench.SetECATTArg("OUTPUTNAME","TestName")
   ```

   The example exports the script argument *TestName* from the .NET script to the script argument *OUTPUTNAME* in eCATT.

   Note: The specified eCATT script argument must match one of the argument names that you have defined in the **eCATT Argument Container**.

Executing Tests in Silk Test Workbench from eCATT

To execute a visual test or a .NET script in eCATT, you have to create and save a BLOB that includes the visual test or the .NET script. When you save a BLOB, the **Primary Test Script/Visual test** setting in the **BLOB Content** area of the **Save BLOB** dialog box is used by eCATT to determine which test to execute.

To execute the script in eCATT that will execute the visual test or .NET script in Silk Test Workbench:

1. If the BLOB that contains the test is not saved, save the BLOB. In Silk Test Workbench, click **Tools** > **SAP eCATT** > **Save BLOB** > **Save and Goto eCATT**. When you save the BLOB, the **Primary Test Script/Visual test** setting in the **BLOB Content** area of the **Save BLOB** dialog box is used by eCATT to determine which test to execute.
2. In eCATT, the **Create Object Directory Entry** dialog box opens. Type the name of the **Package** and click **Save.** The script is now ready to be executed in the eCATT UI.
3. Click **Execute** in the eCATT UI. The **Start Options** dialog box opens.
4. In the **External Tool** area, type the **User Name** and the **Password** for the Silk Test Workbench database.
5. *Optional:* If you want to see what happens in the UI of Silk Test Workbench while the test is being executed, select **With Surface of External Tool** from the **Mode for Ext. Tool** list box.

   When you select **Debug Mode** from the list box, the script breaks at the first line.
6. Click **Run**. eCATT starts Silk Test Workbench, sends the information to it, and executes the script.

Viewing Silk Test Workbench Test Results in eCATT

In eCATT, you can view the test results that are generated by executing a visual test or a .NET script in Silk Test Workbench. When eCATT executes a visual test or a .NET script in Silk Test Workbench, a log is created in eCATT. The log includes the results of the Silk Test Workbench test.

Note: The log does not contain any screenshots that are included in the Silk Test Workbench test results.

Windows API-Based Application Support

Silk Test Workbench provides built-in support for testing Microsoft Windows API-based applications. Several objects exist in Microsoft applications that Silk Test Workbench can better recognize if you enable Accessibility. For example, without enabling Accessibility Silk Test Workbench records only basic information about the menu bar in Microsoft Word and the tabs that appear in Internet Explorer versions later than version 7.0. However, with Accessibility enabled, Silk Test Workbench fully recognizes those objects. You can also improve Silk Test Workbench object recognition by defining a new window, if necessary.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Supported Controls

For a complete list of the record and replay controls available for Windows-based testing, see *Win32 Class Reference*.

Win32 Class Reference

When you configure a Win32 application, Silk Test Workbench automatically provides built-in support for testing standard Windows API-based controls.

Attributes for Windows API-based Client/Server Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Windows API-based client/server applications include:

- caption
- windowid
- priorlabel: Helps to identify text input fields by the text of its adjacent label field. Every input field of a form usually has a label that explains the purpose of the input. For controls that do not have a caption, the attribute **priorlabel** is automatically used in the locator. For the **priorlabel** value of a control, for example a text box, the caption of the closest label at the left side or above the control is used.

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Determining the priorLabel in the Win32 Technology Domain

To determine the priorLabel in the Win32 technology domain, all labels and groups in the same window as the target control are considered. The decision is then made based upon the following criteria:

- Only labels either above or to the left of the control, and groups surrounding the control, are considered as candidates for a priorLabel.
- In the simplest case, the label closest to the control is used as the priorLabel.
- If two labels have the same distance to the control, the priorLabel is determined based upon the following criteria:
  - If one label is to the left and the other above the control, the left one is preferred.
  - If both levels are to the left of the control, the upper one is preferred.
  - If both levels are above the control, the left one is preferred.
- If the closest control is a group control, first all labels within the group are considered according to the rules specified above. If no labels within the group are eligible, then the caption of the group is used as the priorLabel.

Testing Embedded Chrome Applications

An embedded Chrome application is a desktop application with an embedded web browser engine that is based on the Chromium core. Such applications enable you to add web browser capabilities to a desktop application. You can create such an app by using for example the Chromium Embedded Framework (CEF) or the Electron framework.

Silk Test Workbench provides full support for testing embedded Chrome applications that allow remote debugging through the `--remote-debugging-port` command line argument. Silk Test Workbench does not support testing embedded Chrome applications that are based on Java, for example Java AWT and Swing applications.

To test an embedded Chrome application with Silk Test Workbench, you have to set the debugging ports for the executable of the application. Start the application from the command line and set the remote debugging port.

- Silk Test Workbench checks if the `-remote-debugging-port` argument is set in the command line arguments of the embedded Chrome application. If the argument is set, Silk Test Workbench automatically sets the **Enable embedded Chrome support** field to the appropriate executable and debugging port.
- If the `-remote-debugging-port` argument is not set in the command line arguments of the embedded Chrome application, you have to manually specify the executable and the port in the **Enable embedded Chrome support** field:

  1. In the Silk Test Workbench UI, select **Edit Options**.
  2. In the **Options** dialog, select the **Advanced** tab.
  3. In the **Enable embedded Chrome support** option, specify the executable and the port as a comma-separated value pair:

     ```
     <application name>.exe=<port number>
     ```

Note: You cannot test embedded Chrome applications that do not allow remote debugging with Silk Test Workbench.

Note: Silk Test Workbench does not support testing non-browser menus of Electron apps.

Example

For example, you can start the application *myApp* from the command line as follows:

```
myApp.exe --remote-debugging-port=9222
```

You can then specify the executable and port in the **Enable embedded Chrome support** option as follows:

```
myApp.exe=9222
```

Microsoft Foundation Class Support

The class ID of a Microsoft Foundation Class (MFC) control might change over time and therefore cannot be used to generate a stable locator. To avoid generating unstable locators, Silk Test Workbench uses the following attributes for the locators:

- The MFC class name, if the Windows class name of the MFC control starts with `Afx:`.
- The Windows class name, if the Windows class name of the MFC control does not start with `Afx:`.

Silk Test Workbench only supports MFC version 140, and only supports the following combinations:

- Release, x86, MBCS
- Release, x86, Unicode
- Debug, x86, MBCS
- Debug, x86, Unicode
- Release, x64, MBCS
- Release, x64, Unicode
- Debug, x64, MBCS
- Debug, x64, Unicode

Note: To execute existing tests with MFC control locators that have been generated with Silk Test Workbench 18.5 or prior, set the OPT_COMPATIBILITY option in the affected test scripts to version 18.5.0 or prior:

```
'VB .NET code
Agent.SetOption("OPT_COMPATIBILITY", "18.5.0")
```

Cross-Browser Testing

With Silk Test Workbench, you can easily verify the functionality of even the most advanced web application across a variety of browsers, with a single, portable visual test or test script. Silk Test Workbench provides leading support for effective and maintainable cross-browser testing with modern web technologies.

One of the main challenges in test automation is to create and maintain test cost effectively. As different browsers behave differently, web application validation is hard to carry out productively. Silk Test Workbench enables you to focus on writing tests, as it handles the following three areas of cross-browser testing:

Built-in synchronization	This enables you to create scripts that run on all supported browsers, without the need to manually synchronize against asynchronous events, which are typical of highly dynamic web applications such as AJAX, or HTML5. Silk Test Workbench supports synchronization modes for HTML or AJAX as well as all major web environments including Apache Flex, Microsoft Silverlight, and HTML5/AJAX. For additional information, see *Page Synchronization for xBrowser*.
Unified object model	Silk Test Workbench enables you to create and maintain a test which runs across a wide range of different browsers. A unified object model across all browsers gives you the ability to focus on a single browser when you create or maintain a test. Silk Test Workbench ensures that the object you interact with is accessible in the same way on all the other browsers, which saves time and enables you to focus on testing rather than on finding workarounds for different browsers.
Out-of-the-box recording of a cross-browser script	Record a script once and replay it in all the other browsers, without any modifications. This significantly reduces the time and effort it takes to create and maintain test scripts. Nothing is simulated - testing is carried out across real browsers, which ensures that the test behaves exactly as it does for your end user.

With Silk Test Workbench, you can replay tests against web applications that use:

- Internet Explorer.
- Mozilla Firefox, both on Microsoft Windows and on macOS.
- Google Chrome, both on Microsoft Windows and on macOS.
- Microsoft Edge.
- Chrome for Android on an Android device.
- Apple Safari, both on macOS and on an iOS device.
- Embedded browser controls.

Note: You can record tests for web applications using one of the following browsers:

- Internet Explorer.
- Microsoft Edge.
- Mozilla Firefox, both on Microsoft Windows and on macOS.
- Google Chrome 50 or later, both on Microsoft Windows and on macOS.
- A mobile browser on a mobile device.

When recording a script for cross-browser testing, Micro Focus recommends using Google Chrome, Mozilla Firefox, or Microsoft Edge, as a script recorded with Silk Test Workbench against Internet Explorer might slightly differ in comparison to a script recorded on one of the other browsers.

Note: Before you record or playback web applications, disable all browser add-ons that are installed in your system. To disable add-ons in Internet Explorer, click **Tools** > **Internet Options**, click the **Programs** tab, click **Manage add-ons**, select an add-on and then click **Disable**.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Sample Applications

To access the Silk Test sample web applications, go to:

- *http://demo.borland.com/InsuranceWebExtJS/*
- *http://demo.borland.com/gmopost*
- *http://demo.borland.com/gmoajax*

Selecting the Browser for Test Replay

You can define the browser that is used for the replay of a test in the following ways:

- If you execute a test from the UI of Silk Test Workbench and the **Select Browser** dialog box displays, the browser selected in the dialog box is used, and Silk Test Workbench ignores which browser is set in the test script.
- If the **Select Browser** dialog box is disabled, because the **Don't show again** is checked, the application configurations in the individual test scripts determine the browser that is used to execute the tests.

  Note: To re-enable the **Select Browser** dialog box, click **Tools** > **Options** and set the **Show Browser Configuration Dialog** option to **Yes**.
- If you execute a script from the command line or from a Continuous Integration (CI) server, specify the connection string in the application configuration of the script.

  To overwrite the browser that is specified in the application configuration, use the *silktest.configurationName* environment variable or the `browser` command-line parameter.
- If you execute a test from Silk Central, create a configuration suite with a configuration for each browser that you want to test. Then specify the appropriate configuration name. For additional information, refer to the *Silk Central Help*.

Examples of setting the browser by using the silktest.configurationName environment variable

- To use Internet Explorer as the browser, type:

```
SET silktest.configurationName=InternetExplorer
STW.exe –script MyScript
```

- To use Microsoft Edge as the browser, type:

```
SET silktest.configurationName=Edge
STW.exe –script MyScript
```

- To use Mozilla Firefox as the browser, type:

```
SET silktest.configurationName=Firefox
STW.exe –script MyScript
```

- To use Google Chrome as the browser, type:

```
SET silktest.configurationName=GoogleChrome
STW.exe –script MyScript
```

- To use Apple Safari on a Mac as the browser, type:

```
SET silktest.configurationName=host=10.0.0.1  - Safari
STW.exe –script MyScript
```

  In this example, the *host* is the Mac, on which you want to test Apple Safari. The host needs to be connected as a remote location to the machine on which Silk Test Workbench is running. For additional information, see *Editing Remote Locations*.
- To use Google Chrome on an Android device as the browser, use a connection string. For example, if the device ID is 11111111 and the device is connected to the remote machine with the IP address 10.0.0.1, type:

```
SET
silktest.configurationName="platformName=Android;deviceName=Mo
```

```
toG3;deviceId=11111111;host=10.0.0.1 - Chrome"
STW.exe –script MyScript
```

- To use Apple Safari on an iOS device as the browser, use a connection string. For example, if the device ID is 11111111 and the device is connected to the remote machine with the IP address 10.0.0.1, type:

  ```
  SET
  silktest.configurationName="platformName=iOS;deviceName=iPad
  mini;deviceId=11111111;host=10.0.0.1"
  STW.exe –script MyScript
  ```

  Additionally, you have to specify the browser in the application configuration.

Tip: For all examples, you can also set the browser by setting the Java System property *-Dsilktest.configurationName* instead of setting the environment variable *silktest.configurationName*. For example, to use Apple Safari on a Mac as the browser, you can also type:

```
-Dsilktest.configurationName=host=10.0.0.1 -
Safari
STW.exe –script MyScript
```

Tip: Open the **Select Browser** dialog box, for example by starting to replay or record from the Silk Test Workbench UI, to see a list of the browsers that are currently available on your system.

Examples of setting the browser by using the browser parameter

- To use Internet Explorer as the browser, type:

  ```
  STW.exe –browser InternetExplorer –script MyScript
  ```
- To use Microsoft Edge as the browser, type:

  ```
  STW.exe –browser Edge –script MyScript
  ```
- To use Mozilla Firefox as the browser, type:

  ```
  STW.exe –browser Firefox –script MyScript
  ```
- To use Google Chrome as the browser, type:

  ```
  STW.exe –browser GoogleChrome –script MyScript
  ```
- To use Apple Safari on a Mac as the browser, type:

  ```
  STW.exe -connectionstring host=10.0.0.1 - Safari –script
  MyScript
  ```

  In this example, the *host* is the Mac, on which you want to test Apple Safari. The host needs to be connected as a remote location to the machine on which Silk Test Workbench is running. For additional information, see *Editing Remote Locations*.
- To use Google Chrome on an Android device as the browser, use a connection string. For example, if the device ID is 11111111 and the device is connected to the remote machine with the IP address 10.0.0.1, type:

  ```
  STW.exe -connectionstring
  "platformName=Android;deviceName=MotoG3;host=10.0.0.1 -
  Chrome" –script MyScript
  ```
- To use Apple Safari on an iOS device as the browser, use a connection string. For example, if the device ID is 11111111 and the device is connected to the remote machine with the IP address 10.0.0.1, type:

  ```
  STW.exe -connectionstring "platformName=iOS;deviceName=iPad
  mini;host=10.0.0.1" –script MyScript
  ```

  Additionally, you have to specify the browser in the application configuration.

Test Objects for xBrowser

Silk Test Workbench uses the following classes to model a web application:

Class	Description
`BrowserApplication`	Exposes the main window of a web browser and provides methods for tabbing.
`BrowserObject`	Represents the base class for all objects that are contained within a `BrowserApplication`.
`BrowserWindow`	Provides access to tabs and embedded browser controls and provides methods for navigating to different pages.
`DomElement`	Exposes the DOM tree of a web application, including frames, and provides access to all DOM attributes. Specialized classes are available for several DOM elements.
`DomButton`	Represents the top-level container for a web page. It exposes the DOM tree through `DomElement`.
`DomCheckBox`	Represents all DOM elements that were specified using the `<input type='checkbox'>` tag.
`DomForm`	Represents all DOM elements that are specified using the `<form>` tag.
`DomLink`	Represents all DOM elements that were specified using the `<a>` tag.
`DomListBox`	Represents all DOM elements that were specified using the `<select>` tag.
`DomRadioButton`	Represents all DOM elements that were specified using the `<input type='radio'>` tag.
`DomTable`	Represents all DOM elements that were specified using the `<table>` tag.
`DomTableRow`	Represents all DOM elements that were specified using the `<tr>` tag.
`DomTextField`	Represents all DOM elements that were specified using one of the following tags: • `<input type='text'>` • `<input type='password'>` • `<input type='file'>` • `<textarea>`

Object Recognition for xBrowser Objects

The xBrowser technology domain supports dynamic object recognition.

Test cases use locator strings to find and identify objects. A typical locator includes a locator name and at least one locator attribute, such as `"//LocatorName[@locatorAttribute='value']"`.

Locator Names

With other technology types, such as Java SWT, locator names are created using the class name of the test object. With xBrowser, the tag name of the DOM element can also be used as locator name. The following locators describe the same element:

1. Using the tag name: `"//a[@href='http://www.microfocus.com']"`

2. Using the class name: `"//DomLink[@href='http://www.microfocus.com']"`

   To optimize replay speed, use tag names rather than class names.

Locator Attributes	All DOM attributes can be used as locator string attributes. For example, the element `<button automationid='123'>Click Me</button>` can be identified using the locator `"//button[@automationid='123']"`.
Recording Locators	Silk Test Workbench uses a built-in locator generator when recording test cases and using the **Identify Object** dialog box. You can configure the locator generator to improve the results for a specific application.

Page Synchronization for xBrowser

Synchronization is performed automatically before and after every method call. A method call is not started and does not end until the synchronization criteria is met.

Note: Any property access is not synchronized.

Synchronization Modes

Silk Test Workbench includes synchronization modes for HTML and AJAX.

Using the HTML mode ensures that all HTML documents are in an interactive state. With this mode, you can test simple Web pages. If more complex scenarios with Java script are used, it might be necessary to manually script synchronization functions, such as:

- `WaitForObject`
- `WaitForProperty`
- `WaitForDisappearance`
- `WaitForChildDisappearance`

The AJAX mode synchronization waits for the browser to be in a kind of idle state, which is especially useful for AJAX applications or pages that contain AJAX components. Using the AJAX mode eliminates the need to manually script synchronization functions (such as wait for objects to appear or disappear, wait for a specific property value, and so on), which eases the script creation process dramatically. This automatic synchronization is also the base for a successful record and playback approach without manual script adoptions.

Troubleshooting

Because of the true asynchronous nature of AJAX, generally there is no real idle state of the browser. Therefore, in rare situations, Silk Test Workbench will not recognize an end of the invoked method call and throws a timeout error after the specified timeout period. In these situations, it is necessary to set the synchronization mode to HTML at least for the problematic call.

Note: Regardless of the page synchronization method that you use, in tests where a Flash object retrieves data from a server and then performs calculations to render the data, you must manually add a synchronization method to your test. Otherwise, Silk Test Workbench does not wait for the Flash object to complete its calculations. For example, in a visual test, use `Delay` and in a script, use `System.Threading.Thread.Sleep(millisecs)`.

Some AJAX frameworks or browser applications use special HTTP requests, which are permanently open in order to retrieve asynchronous data from the server. These requests may let the synchronization hang until the specified synchronization timeout expires. To prevent this situation, either use the HTML synchronization mode or specify the URL of the problematic request in the **Synchronization exclude list** setting.

To add the URL to the exclusion filter, specify the URL in the **Synchronization exclude list** in the **Options** dialog box.

Use a monitoring tool to determine if playback errors occur because of a synchronization issue. For instance, you can use FindBugs, *http://findbugs.sourceforge.net/*, to determine if an AJAX call is affecting playback. Then, add the problematic service to the **Synchronization exclude list**.

Note: If you exclude a URL, synchronization is turned off for each call that targets the URL that you specified. Any synchronization that is needed for that URL must be called manually. For example, you might need to manually add `WaitForObject` to a script or visual test. To avoid numerous manual calls, exclude URLs for a concrete target, rather than for a top-level URL, if possible.

Configuring Page Synchronization Settings

You can configure page synchronization settings for each individual visual test or script or you can set global options that apply to all visual tests and scripts in the **Options** dialog box. Choose **Tools** > **Options** expand **Playback** and click **xBrowser** to configure these options.

To configure individual settings for visual tests, record the visual test and then insert a **Playback setting** step to override the global playback value. Settings include: **Synchronization exclude list**, **Synchronization mode**, and **Synchronization timeout**.

For example, for visual tests, you might set the **Synchronization mode** playback setting to **HTML** and then return the **Synchronization mode** to **AJAX** for the remaining steps if necessary.

To configure individual settings for scripts, call any of the following:

- `Agent.SetOption(Options.XBrowserSynchronizationMode, mode)`

  For the *mode*, type 1 to use HTML and 2 to use AJAX.
- `Agent.SetOption(Options.XBrowserSynchronizationExcludes, URL)`

  Specify the *URL* or a fragment of the URL for any service or Web page that you want to exclude. For example, you might type `http://example.com/syncsample/timeService`.
- `Agent.SetOption(Options.XBrowserSynchronizationTimeout, time)`

  Specify the *time* in milliseconds.

Comparing API Playback and Native Playback for xBrowser

Silk Test Workbench supports API playback and native playback for Web applications. If your application uses a plug-in or AJAX, use native user input. If your application does not use a plug-in or AJAX, we recommend using API playback.

Advantages of native playback include:

- With native playback, the agent emulates user input by moving the mouse pointer over elements and pressing the corresponding elements. As a result, playback works with most applications without any modifications.
- Native playback supports plug-ins, such as Flash and Java applets, and applications that use AJAX, while high-level API recordings do not.

Advantages of API playback include:

- With API playback, the Web page is driven directly by DOM events, such as `onmouseover` or `onclick`.
- Scripts that use API playback do not require that the browser be in the foreground.
- Scripts that use API playback do not need to scroll an element into view before clicking it.
- Generally API scripts are more reliable since high-level user input is insensitive to pop-up windows and user interaction during playback.
- API playback is faster than native playback.

You can use the **Options** dialog box to configure the types of functions to record and whether to use native user input.

Differences Between API and Native Playback Functions

The `DomElement` class provides different functions for API playback and native playback.

The following table describes which functions use API playback and which use native playback.

	API Playback	Native Playback
Mouse Actions	`DomClick`	`Click`
	`DomDoubleClick`	`DoubleClick`
	`DomMouseMove`	`MoveMouse`
		`PressMouse`
		`ReleaseMouse`
Keyboard Actions	not available	`TypeKeys`
Specialized Functions	`Select`	not available
	`SetText`	
	etc.	

Setting Mouse Move Preferences

Specify whether mouse move actions are recorded for Web applications, Win32 applications, and Windows Forms applications that use mouse move events. You cannot record mouse move events for child domains of the xBrowser technology domain, for example Apache Flex and Swing.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **General**.
4. To record mouse move actions, select **Yes** from the **Record mouse move actions** list box.

   Select **Yes** if you are testing a web page that uses mouse move events.

   Silk Test Workbench will only record mouse move events that cause changes to the hovered element or its parent in order to keep scripts short.
5. If you record mouse move actions, in the **Record mouse move delay** text box, specify how many milliseconds the mouse has to be motionless before a `MoveMouse` action is recorded

   By default this value is set to 200.

   Mouse move actions are only recorded if the mouse stands still for this time. A shorter delay will result in more unexpected move mouse actions, a longer delay will require you to keep the mouse still to record an action.
6. Click **OK**.

Browser Configuration Settings for xBrowser

Several browser settings help to sustain stable test executions. Although Silk Test Workbench works without changing any settings, there are several reasons that you might want to change the browser settings.

Increase replay speed	Use `about:blank` as home page instead of a slowly loading Web page.
Avoid unexpected behavior of the browser	• Disable pop up windows and warning dialog boxes. • Disable auto-complete features. • Disable password wizards.
Prevent malfunction of the browser	Disable unnecessary third-party plugins.

The following sections describe where these settings are located in the corresponding browser.

Internet Explorer

The browser settings are located at **Tools** > **Internet Options**. The following table lists options that you might want to adjust.

Tab	Option	Configuration	Comments
General	Home page	Set to `about:blank`.	Minimize start up time of new tabs.
General	Tabs	• Disable warning when closing multiple tabs. • Enable to switch to new tabs when they are created.	• Avoid unexpected dialog boxes. • Links that open new tabs might not replay correctly otherwise.
Privacy	Pop-up blocker	Disable pop up blocker.	Make sure your Web site can open new windows.
Content	AutoComplete	Turn off completely	• Avoid unexpected dialog boxes. • Avoid unexpected behavior when typing keys.
Programs	Manage add-ons	Only enable add-ons that are absolutely required.	• Third-party add-ons might contain bugs. • Possibly not compatible to Silk Test Workbench.
Advanced	Settings	• Disable **Automatically check for Internet Explorer updates**. • Enable **Disable script debugging (Internet Explorer)**. • Enable **Disable script debugging (Other)**. • Disable **Enable automatic crash recovery**. • Disable **Display notification about every script error**. • Disable all **Warn ...** settings	Avoid unexpected dialog boxes.

Note: Recording a Web application in Internet Explorer with a zoom level different to 100% might not work as expected. Before recording actions against a Web application in Internet Explorer, set the zoom level to 100%.

Mozilla Firefox

You do not have to change browser settings for Mozilla FirefoxSilk Test Workbench automatically starts Mozilla Firefox with the appropriate command-line parameters.

Note: To avoid unexpected behavior when testing web applications, disable auto updates for Mozilla Firefox. For additional information, see *Stop automatic updates*.

Google Chrome

You do not have to change browser settings for Google Chrome. Silk Test Workbench automatically starts Google Chrome with the appropriate command-line parameters.

Note: To avoid unexpected behavior when testing web applications, disable auto updates for Google Chrome. For additional information, see *Turning Off Auto Updates in Google Chrome*.

Connection String for a Remote Desktop Browser

The *connection string* specifies the remote desktop browser that is used for testing. When testing a web application in a remote browser, Silk Test Workbench uses the connection string to connect to the remote location. The connection string is typically part of the application configuration. You can set the connection string when you configure the web application that you want to test. To change the connection string, you can use the **Edit Application Configuration** dialog box.

When testing a web application in a remote browser, the connection string includes only the host, which means the IP address or the host name of the remote machine, for example 10.0.0.1. To select the correct browser, Silk Test Workbench uses the connection string in combination with the browser type, which you can also specify in the **Edit Application Configuration** dialog box.

The host name is case-insensitive.

Note: Remote desktop browser testing is only supported for Microsoft Edge on a remote Microsoft Windows machine, and for Apple Safari on a remote Mac.

Connection string example

```
"host=10.0.0.1"
```

Testing Browsers on a Remote Windows Machine

To test Microsoft Edge, Google Chrome, or Mozilla Firefox on a remote Windows machine, Silk Test Workbench needs to be installed on the remote machine. Microsoft Edge does only run on a Microsoft Windows 10 machine. Silk Test Workbench supports testing Mozilla Firefox 55 or prior on a remote Windows machine.

1. On the local machine, from which you want to test the browser, add the remote Windows machine as a remote location.

   For additional information, see *Editing Remote Locations*.
2. If the Silk Test information service is already running with administrator privileges on the remote machine, disable the Silk Test information service.
   a) Sign in to the remote machine as an administrator.
   b) Open the **Control Panel** (icons view).
   c) Click **Administrative Tools**.
   d) Double-click **Services**.
   e) Double click on the Silk Test information service.
   f) If the service shows a status of running, click **Stop** and wait until the service status shows as `stopped`.
   g) Change the **Startup type** to **Disabled**.
   h) Click **OK**.
3. Start the Silk Test information service with medium integrity level, which means without administrator privileges.

a) Open a file explorer and navigate to `%OPEN_AGENT_HOME%/InfoService`.

   For example, `C:\Program Files (x86)\Silk\SilkTest\ng\InfoService`.

b) Double-click `infoservice_start.bat`.

4. You can now select the browser on the remote machine in the **Select Application** dialog box.

Note: Ensure that the Silk Test information service is running in a user session, and not in the services session. The Silk Test information service requires UI interaction to be enabled. The services session, which is session 0, does not enable UI interaction.

Testing Google Chrome or Mozilla Firefox on a Mac

To test Google Chrome or Mozilla Firefox on a remote macOS machine, the Silk Test information service needs to be installed on the remote machine. For additional information, see *Installing the Silk Test Information Service on a Mac*.

1. On the local machine, from which you want to test Google Chrome or Mozilla Firefox, add the remote macOS machine as a remote location.

   For additional information, see *Editing Remote Locations*.

2. You can now select Google Chrome or Mozilla Firefox on the remote macOS machine in the **Select Application** dialog box.
3. To use the STW.EXE command line to execute a script on a remote browser on the macOS machine, use the following command:

   ```
   stw -d empty -browser <browsername> -connectionstring <connectionstring> -s <Scriptname>
   ```

   For example:

   ```
   stw -d empty -browser Firefox -connectionstring "host=try-imac" -s BrowserScript
   ```

   or

   ```
   stw -d empty -verbose -browser GoogleChrome -connectionstring "host=try-imac" -s BrowserScript
   ```

Note: Ensure that the Silk Test information service is running in a user session, and not in the services session. The Silk Test information service requires UI interaction to be enabled. The services session, which is session 0, does not enable UI interaction.

For information on the prerequisites and limitations when testing on Google Chrome, refer to the topics in the section *Testing with Google Chrome*. For information on the prerequisites and limitations when testing on Mozilla Firefox, refer to the topics in the section *Testing with Mozilla Firefox*.

Setting Capabilities for WebDriver-Based Browsers

If you are testing a web application on a WebDriver-based browser, you can customize and configure the browser session by setting the capabilities.

In Silk Test Workbench, you can specify WebDriver capabilities in the connection string for the following browser types:

- Google Chrome
- Mozilla Firefox

For information on the available options and capabilities for Mozilla Firefox 48 or later, see *https://github.com/mozilla/geckodriver*. For information on the available options and capabilities for Google Chrome, see *Capabilities & ChromeOptions*.

To set the capabilities in Silk Test Workbench:

1. Select the project which corresponds to the web application for which you want to change the capabilities.
2. Edit the connection string in the base state of the project.

   You can edit the connection string in the following ways:

   - By using the **Edit Application Configurations** dialog, for example if you want to record actions against a customized browser.
   - In a script, if you only want to execute the tests in the script against the customized browser.

   For additional information, see *Base State*.
3. Execute the script to start the browser with the specified options and capabilities.

Examples

You can add the following code to the base state in a script to automatically download executables from Mozilla Firefox:

```
' VB .NET code - No linebreaks, this is a single line of code.
 baseState.ConnectionString = "moz:firefoxOptions={\"prefs\":
{ \"browser.download.folderList\": 2,
\"browser.helperApps.neverAsk.saveToDisk\": \"application/octet-
stream\"}};"
```

You can add the following code to the base state in a script to specify the download folder for Mozilla Firefox:

```
' VB .NET code
baseState.ConnectionString = "moz:firefoxOptions={\"prefs\":
{ \"browser.download.dir\" : \"C:/Download\"}};"
```

You can add the following code to the base state in a script to set a command line argument for Mozilla Firefox:

```
' VB .NET code
baseState.ConnectionString = "moz:firefoxOptions={\"args\":[\"--
devtools\"]};")
```

You can add the following code to the base state in a script to automatically download executables from Google Chrome to a specific folder:

```
' VB .NET code - No linebreaks, this is a single line of code.
baseState.ConnectionString = "chromeOptions={\"prefs\":
{\"profile.default_content_setting_values.automatic_downloads\":
1, \"download.default_directory\":\"c:/Download\",
\"download.prompt_for_download\":false}};"
```

You can add the following code to the base state in a script to disable the password manager from showing messages in Google Chrome:

```
' VB .NET code - No linebreaks, this is a single line of code.
baseState.ConnectionString = "chromeOptions={\"args\": [\"--
disable-save-password-bubble\"], \"prefs\":
{\"profile.password_manager_enabled\":false,
\"credentials_enable_service\":false}};"
```

Testing with Apple Safari on a Mac

This section describes how you can enhance your cross-browser test set by testing Apple Safari on Mac machines that are connected to a Windows machine on which Silk Test Workbench is installed.

Prerequisites for Testing with Apple Safari on a Mac

Before you can test with Apple Safari on a Mac, ensure that the following prerequisites are met:

- The Mac is connected as a remote location to a Windows machine, on which Silk Test Workbench is installed. For additional information, see *Editing Remote Locations*.
- If you are testing with Apple Safari 9, the SafariDriver, which is the WebDriver extension for Apple Safari that inverts the traditional client/server relationship and communicates with the WebDriver client using WebSockets, needs to be installed on the Mac. With Apple Safari 10.1, Safari features a built-in driver implementation.
- Java JDK is installed on the Mac.
- The information service is installed on the Mac. To get the files that are required for the information service, use the Silk Test installer. For additional information, see *Installing the Silk Test Information Service on a Mac*.
- To run tests on Apple Safari, the user that has installed the information service needs to be logged in on the Mac.

  Tip: Micro Focus recommends to set the Mac to automatically log in the correct user during startup. For additional information, see *Set your Mac to automatically log in during startup*.
- To run unattended tests against Apple Safari on a Mac, adjust the following energy-related settings in the **Energy Saver** pane of the **System Preferences**:

  - Set **Turn display off after** to **Never**.
  - Check the **Prevent computer from sleeping automatically when the display is off** check box.

  Note: You can use the **Silk Test Configuration Assistant** to easily configure such settings. To open the **Configuration Assistant** on a Mac, click on the Silk Test icon in the status menus and select **Configuration Assistant**.
- To run unattended tests against Apple Safari on a Mac, disable the screen saver.

  1. Navigate to **System Preferences** > **Desktop & Screen Saver**.
  2. Click the **Screen Saver** tab.
  3. Set **Start screen saver** to **Never**.

  Note: You can use the **Silk Test Configuration Assistant** to easily configure such settings. To open the **Configuration Assistant** on a Mac, click on the Silk Test icon in the status menus and select **Configuration Assistant**.
- If you are testing with Apple Safari 10.1, enable the Safari developer menu. Choose **Safari** > **Preferences** , click **Advanced**, and check **Show develop menu in menu bar**.
- If you are testing with Apple Safari 10.1, enable remote automation. In the Safari developer menu, check **Allow Remote Automation**.
- When executing a test for the first time against Apple Safari 10.1, you need to provide a password.

Preparing Apple Safari for Testing

To test web applications on Apple Safari 10.1 or later, you can use the **Silk Test Configuration Assistant** to easily configure Apple Safari. To open the **Configuration Assistant** on a Mac, click on the Silk Test icon in the status menus and select **Configuration Assistant**. As an alternative, you can also perform the following preparation steps in addition to fulfilling the requirements listed in *Prerequisites for Testing with Apple Safari on a Mac*:

1. Enable remote automation in Apple Safari, by opening the **Develop** menu and checking **Allow Remote Automation**.

   The **Develop** menu is hidden by default. To open the menu:

   a) In the **Safari** menu, choose **Preferences**.
   b) In the **Preferences** window, select the **Advanced** tab.
   c) Check the **Show Develop menu in menu bar** check box.
   d) Close the **Preferences** window.
2. When running a test for the first time on Apple Safari, a dialog box appears, stating that the browser window is remotely controlled by an automated test. Click **Continue Session**.

For additional information on Apple Safari and Selenium WebDriver, see *https://webkit.org/blog/6900/webdriver-support-in-safari-10/*.

Installing the Silk Test Information Service on a Mac

Note: To install the information service on a Mac, you require administrative privileges on the Mac.

To create and execute tests against Apple Safari on a Mac, or against mobile applications on an iOS or Android device that is connected to a Mac, install the Silk Test information service (information service) on the Mac, and then use the **Remote Locations** dialog box to connect a Windows machine, on which Silk Test Workbench is installed, to the Mac.

To install the information service on a Mac:

1. Ensure that a Java JDK is installed on the Mac.
2. If you want to test mobile applications on an iOS device, ensure that Xcode is installed on the Mac.
3. Access the information service setup file, `SilkTestInformationService<Version>-<Build Number>.pkg`.
   - If you have downloaded the information service setup file while installing Silk Test, open the folder `macOS` in the Silk Test installation directory, for example `C:\Program Files (x86)\Silk\SilkTest`.
   - If you have not downloaded the information service setup file while installing Silk Test, you can download the setup file from *Micro Focus SupportLine*.
4. Copy the file `SilkTestInformationService<Version>-<Build Number>.pkg` to the Mac.
5. Execute `SilkTestInformationService<Version>-<Build Number>.pkg` to install the information service.
6. Follow the instructions in the installation wizard.
7. When asked for the password, provide the password of the currently signed in Mac user.
8. When Apple Safari opens and a message box asks whether to trust the SafariDriver, click **Trust**.

   Note: You can only install the SafariDriver if you are directly logged in to the Mac, and not connected through a remote connection.

To complete the installation, the installer logs the current Mac user out. To verify that the information service was installed correctly, log in to the Mac and click on the Silk Test icon in the top-right corner of the screen to see the available devices and browsers.

Tip: If the Silk Test icon does not appear, restart the Mac.

Limitations for Testing with Apple Safari

The following are the known limitations for testing with Apple Safari on a Mac:

- The following classes, interfaces, methods, and properties are currently not supported when testing web applications with Apple Safari on a Mac:
  - `BrowserApplication` class.
    - `ClearCache` method
    - `CloseOtherTabs` method
    - `CloseTab` method
    - `ExistsTab` method
    - `GetHorizontalScrollbar` method
    - `GetNextCloseWindow` method
    - `GetSelectedTab` method

    - `GetSelectedTabIndex` method
    - `GetSelectedTabName` method
    - `GetTabCount` method
    - `GetVerticalScrollbar` method
    - `IsActive` method
    - `Minimize` method
    - `OpenContextMenu` method
    - `OpenTab` method
    - `Restore` method
    - `SelectTab` method
    - `SetActive` method
    - `WindowState` property
  - `BrowserWindow` class.
    - `AcceptAlert` method
    - `DismissAlert` method
    - `GetAlertText` method
    - `IsAlertPresent` method
    - `MouseMove` method
    - `PressKeys` method
    - `PressMouse` method
    - `ReleaseKeys` method
    - `ReleaseMouse` method
  - `IMoveable` class.
    - `GetFocus` method.
- Silk Test Workbench does not support the **CMD** key for the `TypeKeys` method.
- Silk Test Workbench does not support testing Apache Flex.
- Silk Test Workbench does not support testing iframes with a JavaScript source on Apple Safari.
- To test secure web applications over HTTPS on Apple Safari, ensure that any required server certificates are trusted.
- Silk Test Workbench does not provide native support for Apple Safari. You cannot test internal Apple Safari functionality. For example, in a test, you cannot change the currently displayed web page by adding text to the navigation bar. As a workaround, you can use API calls to navigate between web pages.
- Silk Test Workbench does not support JavaScript dialog API functions for Apple Safari. As a workaround, you could patch such functions so that they are ignored. For additional information, see *https://groups.google.com/forum/#!topic/selenium-developer-activity/qsovJw93g9c*.
- Silk Test Workbench does not support text recognition with Apple Safari.

  Text recognition includes the following methods:

  - `TextCapture`
  - `TextClick`
  - `TextExists`
  - `TextRectangle`
- Silk Test Workbench does not support tabbing on Apple Safari.
- If the replay of a visual test is slow, set the option **Control capture** under **Tools** > **Options** > **Playback** > **Results** > **Visual test**, to **No**. Otherwise, the playback might get very slow for rich Web pages, and might even result in a communication timeout.
- To test a multi window application, disable the Apple Safari pup-up blocker. To do so, start Apple Safari and navigate to **Safari Preferences** > **Security** > **Block pop-up window**.

- Silk Test Workbench does not support testing the dialog box for saving a password. To avoid this dialog box, start Apple Safari, navigate to **Safari Preferences** > **AutoFill**, and check the **User names and passwords** check box.
- Silk Test Workbench does not support properties in XPath expressions for Apple Safari. Only attributes are supported in XPath expressions.
- Silk Test Workbench does not support testing web applications which include a Content-Security-Policy HTTP header.
- With Apple Safari 10.1, Silk Test Workbench does not support navigating back in the browser.
- With Apple Safari 10.1, Silk Test Workbench does not support using control keys in the `TypeKeys` method.
- With Apple Safari 10.1, Silk Test Workbench only supports dom actions in Frames and IFrames.
- With Apple Safari 10.1, Silk Test Workbench does not support navigating with Frames and IFrames.
- With Apple Safari 10.1, Silk Test Workbench does not support direct scrolling during recording. As a workaround, you could use the executeJavaScript method.

Running Multiple Apple Safari Tests at the Same Time

To execute a test on Apple Safari, you require a Mac that is connected to the Windows machine on which Silk Testis installed. If multiple Apple Safari want to execute tests on Apple Safari, these tests can be executed simultaneously on the same Mac.

Note: Each test that is executed against Apple Safari on the Mac opens an individual instance of Apple Safari. Having too many instances of Apple Safari running simultaneously might reduce the performance of the Mac.

Uninstalling the Silk Test Information Service from a Mac

To uninstall the Silk Test information service (information service) from a Mac, for example if you no longer want to execute tests against Apple Safari on the Mac:

1. Create a new shell file, for example `uninstallInfoService.sh`.
2. Type the following code into the new file:

```
#!/bin/sh

if launchctl list | grep com.borland.infoservice ; then
  launchctl unload /Library/LaunchAgents/com.borland.infoservice.plist
  echo "unloading Launch Daemon"
fi

if [ -d "/Applications/Silk" ]
then
  sudo rm -rf /Applications/Silk
fi

if [ -f "/Library/LaunchAgents/com.borland.infoservice.plist" ]
then
  sudo rm /Library/LaunchAgents/com.borland.infoservice.plist
fi

if [ -f "/usr/local/bin/ideviceinstaller" ]
then
  sudo rm /usr/local/bin/ideviceinstaller
fi

exit 0
```

3. In the command line, type `chmod +x uninstallInfoService.sh` to make the shell file executable.
4. Execute the shell file from the command line.

Testing with Google Chrome

This section describes how you can enhance your cross-browser test set by testing with Google Chrome.

Silk Test Workbench supports recording actions and replaying tests on Google Chrome 50 or later, and supports replaying tests and recording locators on Google Chrome versions prior to version 50.

- When starting to test on Google Chrome 50 or later, with no running instance of Google Chrome open, Silk Test Workbench starts a new instance of Google Chrome. This new browser uses a temporary profile without add-ons and with an empty cache.
- When starting to test on an instance of Google Chrome 50 or later, which is already running, Silk Test Workbench restarts Google Chrome with the same command line arguments that were used when the instance was initially started. This restart is required to enable the Silk Test Workbench automation support.
- When testing with Google Chrome 50 or later, the Google Chrome instance is closed when shutting down the Open Agent or when starting to test another application outside Google Chrome.

Tip: If you want to execute an existing test script with Google Chrome 50 or later, Micro Focus recommends that you use a base state or that you add a command to the test script to navigate to the URL.

Example 1

If the running instance of Google Chrome 50 or later was initially started with the command `C:/Program Files (x86)/Google/Chrome/Application/chrome.exe www.borland.com`, Google Chrome opens to *www.borland.com* after the restart.

Example 2

If the running instance of Google Chrome 50 or later was initially started with the command `C:/Program Files (x86)/Google/Chrome/Application/chrome.exe`, Google Chrome opens to *about:blank* after the restart.

Prerequisites for Replaying Tests with Google Chrome

Command-line parameters

When you use Google Chrome to replay a test or to record locators, Google Chrome is started with the following command:

```
%LOCALAPPDATA%\Google\Chrome\Application\chrome.exe
  --enable-logging
  --log-level=1
  --disable-web-security
  --disable-hang-monitor
  --disable-prompt-on-repost
  --dom-automation
  --full-memory-crash-report
  --no-default-browser-check
  --no-first-run
  --homepage=about:blank
  --disable-web-resources
  --disable-preconnect
  --enable-logging
  --log-level=1
  --safebrowsing-disable-auto-update
  --test-type=ui
  --noerrdialogs
```

```
  --metrics-recording-only
  --allow-file-access-from-files
  --disable-tab-closeable-state-watcher
  --allow-file-access
  --disable-sync
  --testing-channel=NamedTestingInterface:st_42
```

When you use the wizard to hook on to an application, these command-line parameters are automatically added to the base state. If an instance of Google Chrome is already running when you start testing, without the appropriate command-line parameters, Silk Test Workbench closes Google Chrome and tries to restart the browser with the command-line parameters. If the browser cannot be restarted, an error message displays.

Note: The command-line parameter `disable-web-security` is required when you want to record or replay cross-domain documents.

Note: To test a web application that is stored in the local file system, navigate to the `chrome://extensions` in Google Chrome and check the **Allow access to file URLs** check box for the **Silk Test Chrome Extension**.

Testing Google Chrome Extensions

You can use one of the following two approaches to test a Google Chrome extension (add-on) with Silk Test Workbench:

Install the extension as a .crx file when starting Google Chrome	To test a Google Chrome extension that is installed as a .crx file, add the following command line to the base state:
	`chrome.exe --load-extension=C:/myExtension/myExtension.crx`
	Note: You can only install a single extension in Google Chrome as a .crx file. To install multiple extensions in Google Chrome, use a comma separated list of .crx files. For example:
	`chrome.exe --load-extension=C:/myExtension/myExtension.crx,C:/myExtension2/myExtension2.crx`
	For information on adding command line arguments to a browser, see *Modifying the Base State*.

Add the extension to a profile	Add the extension to a Google Chrome user data directory and use that profile for testing. For additional information, see *Testing Google Chrome with User Data Directories*.

Testing Google Chrome with User Data Directories

All changes that you make in Google Chrome, for example your home page, what toolbars you use, any saved passwords, and your bookmarks, are all stored in a special folder, which is called a user data directory.

With Silk Test Workbench, you can test Google Chrome user data directories by specifying the path to the user data directory in the base state of the application under test. The following command line includes the path to the profile:

```
chrome.exe "--user-data-dir=C:/Users/MyUser/AppData/Local/Google/Chrome/User Data"
```

To set the profile directory for our sample web application, you can use the following code:

```
' VB code
Dim baseState = New BrowserBaseState(BrowserType.GoogleChrome,
"demo.borland.com/InsuranceWebExtJS")
Dim myProfileDir = "--user-data-dir=D:\\temp\\SilkTest --profile-
directory=Profile1"
```

```
baseState.CommandLineArguments = myProfileDir
baseState.Execute()
```

Note: When Google Chrome is started by Silk Test Workbench, an empty user data directory is used. This ensures that the test starts at a clean state.

Limitations for Testing with Google Chrome

The following list lists the known limitations for playing back tests and recording locators with Google Chrome on a local Windows machine:

- Silk Test does not support testing child technology domains of the xBrowser domain with Google Chrome. For example Apache Flex or Microsoft Silverlight are not supported with Google Chrome.
- Silk Test Workbench does not support recording a test in an HTTP Basic Authentication dialog.
- Silk Test does not provide native support for Google Chrome. You cannot test internal Google Chrome functionality. For example, in a test, you cannot change the currently displayed web page by adding text to the navigation bar through Win32. As a workaround, you can use API calls to navigate between web pages. Silk Test supports handling alerts and similar dialog boxes through the Alerts API.
- Silk Test Workbench does not support text recognition with Google Chrome.

  Text recognition includes the following methods:

  - `TextCapture`
  - `TextClick`
  - `TextExists`
  - `TextRectangle`
- Silk Test Workbench does not support enumerations when testing Web applications with Google Chrome.
- Silk Test Workbench does not support the `GetFocus` method of the `IMoveable` class.
- Silk Test does not recognize opening the **Print** dialog box in Google Chrome by using the Google Chrome menu. To add opening this dialog box in Google Chrome to a test, you have to send **Ctrl+Shift+P** using the `TypeKeys` method. Internet Explorer does not recognize this shortcut, so you have to first record your test in Internet Explorer, and then manually add pressing **Ctrl+Shift+P** to your test.
- Testing on multiple Google Chrome windows at the same time is only supported if the additional windows are opened from the initial Google Chrome window by the AUT itself. If the additional Google Chrome windows are opened manually, Silk Test Workbench does not recognize the elements on these Google Chrome windows. For example, Silk Test Workbench recognizes the elements in a Google Chrome window that is opened by clicking on a link or a button in the AUT during recording, but Silk Test Workbench does not recognize the elements in a Google Chrome window that was opened by pressing **CTRL+N** during recording.
- If the replay of a visual test is slow, set the option **Control capture** under **Tools** > **Options** > **Playback** > **Results** > **Visual test**, to **No**. Otherwise, the playback might get very slow for rich Web pages, and might even result in a communication timeout.
- With Google Chrome 49 or prior and when using Internet Explorer to replay a test, you can use the following code to test executeJavaScript:

  ```
  // Java code
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("function foo() { alert('Silk Test'); }");
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("foo();");
  ```

  When replaying tests on Google Chrome, the scripts are not executed in the global context (window), but in a closure. Everything is executed within a function. The first `ExecuteJavaScript` call in the previous code sample will not work with Google Chrome, because the function `foo` is only available as long as the `ExecuteJavaScript` call lasts.

  To replay the same test on Google Chrome, you can use the following function expression:

  ```
  // Java code
  desktop.<BrowserWindow> find("//BrowserWindow")
  ```

```
  .executeJavaScript("window.foo = function() { alert('Silk Test'); }");
desktop.<BrowserWindow> find("//BrowserWindow")
  .executeJavaScript("window.foo();");
```

The previous code samples will work in Silk4J. The code for the other Silk Test clients is similar. For additional information, refer to the documentation of the `ExecuteJavaScript` method in the Help of your Silk Test client.

Limitations for Testing with Google Chrome on macOS

The following list lists the known limitations for playing back tests and recording locators with Google Chrome on macOS:

- Silk Test Workbench does not support the **CMD** key for the `TypeKeys` method.
- Silk Test does not support testing child technology domains of the xBrowser domain with Google Chrome. For example Apache Flex or Microsoft Silverlight are not supported with Google Chrome.
- Silk Test Workbench does not support recording a test in an HTTP Basic Authentication dialog.
- Silk Test does not provide native support for Google Chrome. You cannot test internal Google Chrome functionality. For example, in a test, you cannot change the currently displayed web page by adding text to the navigation bar through Win32. As a workaround, you can use API calls to navigate between web pages. Silk Test supports handling alerts and similar dialog boxes through the Alerts API.
- Silk Test Workbench does not support text recognition with Google Chrome.

  Text recognition includes the following methods:

  - `TextCapture`
  - `TextClick`
  - `TextExists`
  - `TextRectangle`
- Silk Test Workbench does not support the `GetFocus` method of the `IMoveable` class.
- Testing on multiple Google Chrome windows at the same time is not supported on macOS.
- Attaching to an already opened Google Chrome window on macOS is not supported.
- With Google Chrome 49 or prior and when using Internet Explorer to replay a test, you can use the following code to test executeJavaScript:

  ```
  // Java code
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("function foo() { alert('Silk Test'); }");
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("foo();");
  ```

  When replaying tests on Google Chrome, the scripts are not executed in the global context (window), but in a closure. Everything is executed within a function. The first `ExecuteJavaScript` call in the previous code sample will not work with Google Chrome, because the function `foo` is only available as long as the `ExecuteJavaScript` call lasts.

  To replay the same test on Google Chrome, you can use the following function expression:

  ```
  // Java code
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("window.foo = function() { alert('Silk Test'); }");
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("window.foo();");
  ```

  The previous code samples will work in Silk4J. The code for the other Silk Test clients is similar. For additional information, refer to the documentation of the `ExecuteJavaScript` method in the Help of your Silk Test client.

Testing with Mozilla Firefox

This section describes how you can enhance your cross-browser test set by testing with Mozilla Firefox.

Silk Test Workbench supports recording actions and replaying tests on Mozilla Firefox.

- When starting to test on Mozilla Firefox, with no running instance of Mozilla Firefox open, Silk Test Workbench starts a new instance of Mozilla Firefox. This new browser uses a temporary profile without add-ons and with an empty cache.
- When starting to test on an instance of Mozilla Firefox which is already running, Silk Test Workbench restarts Mozilla Firefox with the same command line arguments that were used when the instance was initially started. This restart is required to enable the Silk Test Workbench automation support.
- The Mozilla Firefox instance is closed when shutting down the Open Agent or when starting to test another application outside Mozilla Firefox.

Tip: If you want to execute an existing test script with Mozilla Firefox, Micro Focus recommends that you use a base state or that you add a command to the test script to navigate to the URL.

While recording with Silk Test Workbench on Mozilla Firefox 52 or later, Mozilla Firefox opens external links in a new tab, instead of a new window. Disable the **Use single-window mode for Firefox** option in the **Advanced** options to open external links in a new window.

Example 1

If the running instance of Mozilla Firefox was initially started with the command `C:/program files/Mozilla/firefox.exe www.borland.com`, Mozilla Firefox opens to *www.borland.com* after the restart.

Example 2

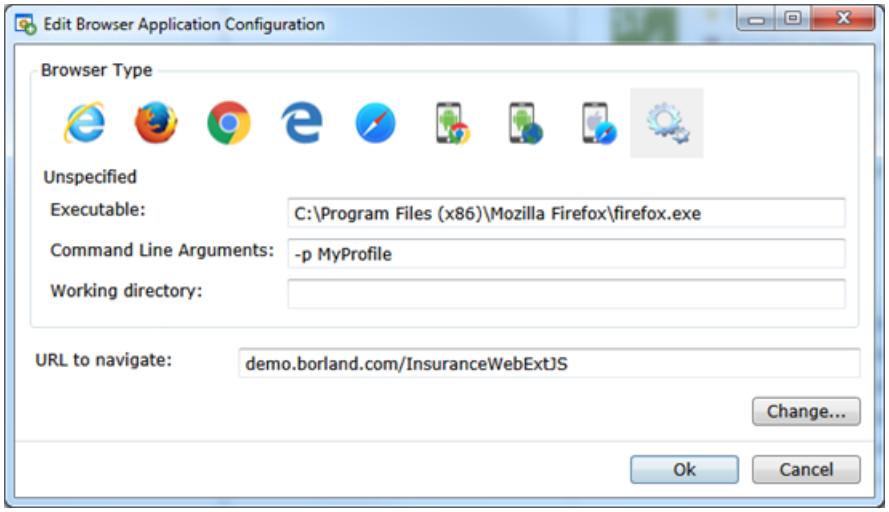
If the running instance of Mozilla Firefox was initially started with the command `C:/program files/Mozilla/firefox.exe`, Mozilla Firefox opens to *about:blank* after the restart.

Testing Mozilla Firefox with Profiles

All changes that you make in Mozilla Firefox, for example your home page, what toolbars you use, any saved passwords, and your bookmarks, are all stored in a special folder, which is called a profile.

To test Mozilla Firefox profiles, you can specify either the name of the profile or the path to the profile as custom browser command line arguments in the **Edit Application Configurations** dialog box:

1. Select the project from which you want to test Mozilla Firefox profiles.
2. Click **Edit Application Configuration** in the menu. The **Edit Application Configuration** dialog box appears.
3. Click **Edit** to edit the existing **Browser: Firefox** application configuration.
4. Select **Custom** as the browser type.
5. Type the path to the Mozilla Firefox executable into the **Executable** field. For example `C:\Program Files (x86)\Mozilla Firefox\firefox.exe`.
6. Type the Firefox profile name or the path to the Firefox profile into the **Command Line Arguments** field. For example `-p myProfile` or `-profile C:/Temp`



Using the name of the profile	The following command line argument specifies the name of the profile: `-p myProfile` You can configure the named profile in the **Profile Manager**. To start the **Profile Manager**, type `firefox.exe -P` into a command window.
Using the path to the profile	The following command line argument specifies the path to the profile: `-profile C:/<path to profile folder>`

Note: If Mozilla Firefox is started by Silk Test Workbench, an empty profile is used. This ensures that the test starts at a clean state. If the user manually starts Mozilla Firefox, the default profile is used.

Note: As profiles need to be deployed to the test machine are and have high memory consumption, you might face some issues when testing profiles. If you only want to change a few browser settings, you can use capabilities instead of profiles. For additional information, see *Setting Capabilities for Web-Driver Based Browsers*.

Testing Mozilla Firefox Extensions

To test a Mozilla Firefox extension (add-on) with Silk Test Workbench, add the extension to a Mozilla Firefox profile and use that profile for testing. For additional information, see *Testing Mozilla Firefox with Profiles*.

Limitations for Testing with Mozilla Firefox

The following limitations are known when testing web applications with Silk Test Workbench on Mozilla Firefox:

- Silk Test Workbench does not support Mozilla Firefox 47, 48, 49, 50, and 51. However, Silk Test Workbench supports Mozilla Firefox 47.0.1 and 47.0.2
- Testing on multiple Mozilla Firefox windows at the same time is only supported if the additional windows are opened from the initial Mozilla Firefox window by the AUT itself. If the additional Mozilla Firefox windows are opened manually, Silk Test Workbench does not recognize the elements on these Mozilla Firefox windows. For example, Silk Test Workbench recognizes the elements in a Mozilla Firefox window that is opened by clicking on a link or a button in the AUT during recording, but Silk Test Workbench does not recognize the elements in a Mozilla Firefox window that was opened by pressing **CTRL+N** during recording.
- Silk Test Workbench does not support testing modal browser windows, which are windows that can be displayed with the `window.showmodaldialog` command. These modal browser windows have been officially deprecated, and are disabled with Google Chrome 37 or later, while they are planned to no longer be supported in future versions of Mozilla Firefox. You can workaround this issue by using low-

level actions, for example native clicks with coordinates to click on an object or typekeys to fill out text fields.

- Silk Test Workbench does not support testing Silverlight with Mozilla Firefox.
- Silk Test Workbench does not support testing some browser dialogs, for example the **About** dialog, with Mozilla Firefox.
- Silk Test Workbench does not support testing `about:*` pages with Mozilla Firefox.
- Silk Test Workbench does not support recording a click on the **Print** button in Mozilla Firefox. To click on this button during replay, you can manually add a desktop click with coordinates to your test script. For example:

  ```
  'VB .NET code
  _desktop.Click(printButton.GetRect(true).Center)
  ```
- Silk Test Workbench does not support text recognition with Mozilla Firefox.

  Text recognition includes the following methods:

  - `TextCapture`
  - `TextClick`
  - `TextExists`
  - `TextRectangle`
- Silk Test does not provide native support for Mozilla Firefox. You cannot test internal Mozilla Firefox functionality. For example, in a test, you cannot change the currently displayed web page by adding text to the navigation bar through Win32. As a workaround, you can use API calls to navigate between web pages. Silk Test supports handling alerts and similar dialog boxes through the Alerts API.
- Silk Test Workbench does not support recording locators on JavaScript alert boxes with Mozilla Firefox. With Mozilla Firefox version 58 or prior, you can use the following methods to handle Javascript alert boxes:

  - `AcceptAlert`
  - `DismissAlert`
  - `GetAlertText`
  - `IsAlertPresent`

  Note: You cannot use these methods with Mozilla Firefox 59 or later.
- Silk Test Workbench does not support Java applets for Mozilla Firefox 52 or later. Silk Test Workbench supports Java applets for Mozilla Firefox 47.0.1 or prior with the following limitations:

  - Silk Test Workbench does not support the locator `//AppletContainer`.
  - When the applet opens a modal dialog, a locator similar to `//BrowserApplication//BrowserWindow/JDialog[@caption='Information']//JButton[@caption='OK']` might not work. You can use a locator like `//JDialog[@caption='Information']//JButton[@caption='OK']` instead.
- Silk Test Workbench does not support properties in XPath expressions for Mozilla Firefox. Only attributes are supported in XPath expressions.
- Silk Test Workbench does not support the `GetFocus` method of the `IMoveable` class.
- Silk Test does not support testing child technology domains of the xBrowser domain with Mozilla Firefox. For example Apache Flex or Microsoft Silverlight are not supported with Mozilla Firefox.
- With Mozilla Firefox 52 or later, the following methods are not supported:

  - `PressKeys`
  - `ReleaseKeys`
- With Mozilla Firefox 52 or prior, the `SetViewportSize` method of the `BrowserWindow` class is not supported.
- With Mozilla Firefox 52 or later, native playback for the following is not supported:

  - Double-click.

    - Right and middle mouse button click.
- With Mozilla Firefox 52 or later, the `DomClick` method is not supported on controls that open an alert.
- With Mozilla Firefox 55, uploading a file does not work. For additional information, see *File upload no longer works with geckodriver 0.18.0 and Firefox 55*.

Limitations for Testing with Mozilla Firefox on macOS

The following limitations are known when testing web applications with Mozilla Firefox on macOS:

- Silk Test Workbench has been tested on macOS with Mozilla Firefox 54 or later.
- Testing on multiple Mozilla Firefox windows at the same time is only supported if the additional windows are opened from the initial Mozilla Firefox window by the AUT itself. If the additional Mozilla Firefox windows are opened manually, Silk Test Workbench does not recognize the elements on these Mozilla Firefox windows. For example, Silk Test Workbench recognizes the elements in a Mozilla Firefox window that is opened by clicking on a link or a button in the AUT during recording, but Silk Test Workbench does not recognize the elements in a Mozilla Firefox window that was opened by pressing **CTRL+N** during recording.
- Silk Test Workbench does not support testing modal browser windows, which are windows that can be displayed with the `window.showmodaldialog` command. These modal browser windows have been officially deprecated, and are disabled with Google Chrome 37 or later, while they are planned to no longer be supported in future versions of Mozilla Firefox. You can workaround this issue by using low-level actions, for example native clicks with coordinates to click on an object or typekeys to fill out text fields.
- Silk Test Workbench does not support testing Silverlight with Mozilla Firefox.
- Silk Test Workbench does not support testing some browser dialogs, for example the **About** dialog, with Mozilla Firefox.
- Silk Test Workbench does not support testing `about:*` pages with Mozilla Firefox.
- Silk Test Workbench does not support recording a click on the **Print** button in Mozilla Firefox. To click on this button during replay, you can manually add a desktop click with coordinates to your test script. For example:

  ```
  'VB .NET code
  _desktop.Click(printButton.GetRect(true).Center)
  ```
- Silk Test Workbench does not support text recognition with Mozilla Firefox.

  Text recognition includes the following methods:

    - `TextCapture`
    - `TextClick`
    - `TextExists`
    - `TextRectangle`
- Silk Test does not provide native support for Mozilla Firefox. You cannot test internal Mozilla Firefox functionality. For example, in a test, you cannot change the currently displayed web page by adding text to the navigation bar through Win32. As a workaround, you can use API calls to navigate between web pages. Silk Test supports handling alerts and similar dialog boxes through the Alerts API.
- Silk Test Workbench does not support recording locators on JavaScript alert boxes with Mozilla Firefox. With Mozilla Firefox version 58 or prior, you can use the following methods to handle Javascript alert boxes:

    - `AcceptAlert`
    - `DismissAlert`
    - `GetAlertText`
    - `IsAlertPresent`

  Note: You cannot use these methods with Mozilla Firefox 59 or later.
- Silk Test Workbench does not support Java applets for Mozilla Firefox on macOS.

- Silk Test Workbench does not support properties in XPath expressions for Mozilla Firefox. Only attributes are supported in XPath expressions.
- Silk Test Workbench does not support the `GetFocus` method of the `IMoveable` class.
- Silk Test does not support testing child technology domains of the xBrowser domain with Mozilla Firefox. For example Apache Flex or Microsoft Silverlight are not supported with Mozilla Firefox.
- The following methods are not supported:
  - `PressKeys`
  - `ReleaseKeys`
- Native playback for the following is not supported:
  - Double-click.
  - Right and middle mouse button click.
- The `DomClick` method is not supported on controls that open an alert.
- With Mozilla Firefox 55, uploading a file does not work. For additional information, see *File upload no longer works with geckodriver 0.18.0 and Firefox 55*.

Testing with Microsoft Edge

This section describes how you can enhance your cross-browser test set by testing with Microsoft Edge.

Limitations for Testing with Microsoft Edge

The following are the known limitations for testing with Microsoft Edge:

- The following classes, interfaces, methods, and properties are currently not supported when testing web applications on Microsoft Edge:
  - `BrowserApplication` class.
    - `ClearCache` method
    - `CloseOtherTabs` method
    - `CloseTab` method
    - `ExistsTab` method
    - `GetHorizontalScrollbar` method
    - `GetNextCloseWindow` method
    - `GetSelectedTab` method
    - `GetSelectedTabIndex` method
    - `GetSelectedTabName` method
    - `GetTabCount` method
    - `GetVerticalScrollbar` method
    - `IsActive` method
    - `Minimize` method
    - `OpenContextMenu` method
    - `OpenTab` method
    - `Restore` method
    - `SelectTab` method
    - `SetActive` method
    - `WindowState` method
  - The following methods of the `BrowserWindow` class are not supported for Microsoft Edge versions prior to build 38.14393, the Microsoft Edge version for Microsoft Windows 10 Anniversary Update.
    - `PressKeys` method
    - `ReleaseKeys` method

- Silk Test Workbench does not automatically bring the browser into the foreground when recording actions against Microsoft Edge.
- When testing with Microsoft Edge, the rectangle for the `BrowserApplication` is not absolute.
- Silk Test Workbench does not support testing Apache Flex.
- Silk Test Workbench does not provide native support for Microsoft Edge. You cannot test internal Microsoft Edge functionality. For example, in a test, you cannot change the currently displayed web page by adding text to the navigation bar. As a workaround, you can use API calls to navigate between web pages.
- Silk Test Workbench does not support handling alerts and similar dialog boxes for Microsoft Edge.
- Image clicks are only supported for Microsoft Edge Threshold 2 (build 25.10586) or later. If you are testing a web application on a prior version of Microsoft Edge, you can only use image verifications.
- Silk Test Workbench does not support text recognition with Microsoft Edge.

  Text recognition includes the following methods:

  - `TextCapture`
  - `TextClick`
  - `TextExists`
  - `TextRectangle`
- Silk Test Workbench does not support tabbing on Microsoft Edge. Tabs are recognized as windows.
- If the replay of a visual test is slow, set the option **Control capture** under **Tools** > **Options** > **Playback** > **Results** > **Visual test**, to **No**. Otherwise, the playback might get very slow for rich Web pages, and might even result in a communication timeout.
- When testing web applications on Microsoft Edge, Silk Test Workbench cannot locate meta-tags which include the `http-equiv` attribute. For example, Silk Test Workbench cannot locate the following meta-tag:

  ```
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  ```
- With Microsoft Edge, Silk Test Workbench does not support directly reading the `currentStyle` attribute of a DOM element. You can use the `GetCssStyle` method of the `DomElement` class to retrieve the computed CSS style with the specified style name.
- When starting the interaction with a web application on Microsoft Edge, Silk Test Workbench closes any open instance of Microsoft Edge and starts a new browser. This new browser uses a temporary profile without add-ons and with an empty cache. This instance of Microsoft Edge is closed when shutting down the Open Agent or when starting to test another application outside Microsoft Edge.
- With Microsoft Edge, Silk Test Workbench does not recognize the `textContents` attribute while recording actions or locators. However, you can use the `textContents` attribute in object maps and when replaying a test on Microsoft Edge.
- Silk Test Workbench does not support properties in XPath expressions for Microsoft Edge. Only attributes are supported in XPath expressions.
- Silk Test Workbench does not support the `GetFocus` method of the `IMoveable` class.
- Silk Test Workbench does not support testing web applications which include a Content-Security-Policy HTTP header.

Responsive Web Design Testing

Desktop web applications which are built based upon responsive web design might change their appearance in response to the size of the screen or web browser in which these applications are displayed. Choosing the appropriate size of the replay window might have significant impact on the stability of such tests.

Silk Test Workbench enables you to specify the exact size of the browser window in the following situations:

- When you create a new project for a web application.
- When you create a new visual test or .NET script.

- When you record a new test.
- When you record actions into an existing test.
- When you replay a test against a web application.

You can use the following settings to specify the size of the browser window:

- The **Browser size** list contains a mix of predefined and custom browser window sizes, enabling you to select the size on which you want to test.
- The **Orientation** list enables you to select whether you want the browser window to use landscape or portrait orientation.
- Click **Edit Browser Sizes** to add custom browser sizes to the **Browser size** list or to specify which browser sizes are displayed in the list.
  - To add a new custom browser size to the list, click **Add Browser Size**.
  - To exclude a size from the list, uncheck the corresponding check box.
  - To add the visual breakpoints of a specific web application to the **Browser size** list, click **Detect Visual Breakpoints**.

When replaying a test against a web application from the command line or from Silk Central, you can specify the size of the browser viewport by setting the *silktest.browserViewportSize* environment variable. You can either specify the name of a browser from the **Browser Size** list or a specific size.

Example: Setting the browser size for automated replay by using the name

The following code sample sets the browser size to SVGA (800, 600) by using the SVGA entry of the **Browser size** list:

```
SET silktest.browserViewportSize=name=SVGA;orientation=landscape
STW.exe –script MyScript
```

You can also specify a specific browser and viewport size by using the following command:

```
STW –dsn silktest –browser InternetExplorer –script MyScript -
viewportname SVGA
```

Example: Setting the browser size for automated replay by using the width and height

The following code sample sets the browser size to SVGA (800, 600) by using the `width` and `height` parameters:

```
SET
silktest.browserViewportSize=width=800;height=600;orientation=la
ndscape
STW.exe –script MyScript
```

You can also specify a specific browser and viewport size by using the following command:

```
STW –dsn silktest –browser InternetExplorer –script MyScript -
viewportwidth 800 -vieportheight 600
```

Detecting Visual Breakpoints

Before detecting the visual breakpoints in a responsive web application, ensure that Mozilla Firefox 56 or later or Google Chrome 60 or later is installed on the machine on which Silk Test Workbench is running.

Many web applications that are implemented with responsive web design techniques change their layout in response to the size of the browser or device in which they are displayed. The specific resolutions on which the layout changes are called *visual breakpoints*.

Silk Test Workbench supports testing such applications by detecting the visual breakpoints, and by allowing you to resize the recording window to the specific size of such a visual breakpoint.

Silk Test Workbench enables you to specify the exact size of the browser window in the following situations:

- When you create a new project for a web application.
- When you create a new visual test or .NET script.
- When you record a new test.
- When you record actions into an existing test.
- When you replay a test against a web application.

To find the visual breakpoints in a web application, and to display the corresponding resolutions in the **Browser size** list, perform the following actions:

1. Click **Edit Browser Sizes**. The **Edit Browser Sizes** dialog appears.
2. Click **Detect Visual Breakpoints**. If no URL for the web application is specified in an application configuration or base state, the **Visual Breakpoint Detection URL** dialog appears.
3. If no URL for the web application has been specified in an application configuration or base state, the **Visual Breakpoint Detection URL** dialog appears. Type the URL into the text field and click **OK**. Silk Test Workbench detects all visual breakpoints for the web application and adds them to the **Browser sizes** list.
4. Click **OK** to close the **Edit Browser Sizes** dialog.

You can now select any of the visual breakpoints as the size of the browser window or mobile device for testing.

Testing Additional Browser Versions

This topic describes how you can test on additional versions of WebDriver-based browsers, which are not automatically included in your Silk Test Workbench version.

Note: Silk Test Workbench should be able to automatically support the newest version of a browser. However, if the browser vendor has introduced changes which require an update to Silk Test Workbench, the procedure described in this topic might not enable testing on the new browser version.

The functionality described in this topic is supported for the following browsers:

- Google Chrome
- Microsoft Edge
- Mozilla Firefox

1. Download the appropriate driver for the browser version that you want to test.
   - For Google Chrome, you can download additional ChromeDriver versions from *Downloads - ChromeDriver*.
   - For Mozilla Firefox, you can download additional geckodriver versions from *Releases - mozilla/geckodriver*.
   - For Microsoft Edge, you can download additional Microsoft WebDriver versions from *WebDriver - Microsoft Edge*.
2. In the Silk Test Workbench installation folder, navigate to the folder `\ng\WebDrivers\`.
3. Open the folder that corresponds to the operating system on which you want to use the new browser version:
   - For Microsoft Windows, open the folder `Windows`.
   - For macOS, open the folder `osx64`.
4. Open the appropriate folder for the browser.

- For Google Chrome, open `Chrome`.
- For Mozilla Firefox, open `Gecko`.
- For Microsoft Edge, open `Edge`.

5. Create a new folder for the new driver version.
   For example, if the driver is ChromeDriver 2.26, create the new folder `2.26`.
6. Extract the downloaded driver into the new folder.
7. In the Silk Test Workbench installation folder, navigate to the folder `\ng\WebDrivers\Common \Config\`.
8. Open the appropriate folder for the browser.
   - For Google Chrome, open `Chrome`.
   - For Mozilla Firefox, open `Gecko`.
   - For Microsoft Edge, open `Edge`.
9. Open the properties file in a text editor.
   For example, for Google Chrome, open `Chrome.properties`.
10. Add the new browser version and the new driver version as follows:
    ```
    <browser version>=<driver version>
    ```
    For example, if you want to test on Google Chrome 53, you require ChromeDriver 2.26 and you have to add the following line to the `Chrome.properties` file:
    ```
    53=2.26
    ```
11. Save the properties file.

Cross-Browser Testing: Frequently Asked Questions

This section includes questions that you might encounter when testing your Web application on various browsers.

Dialog is Not Recognized During Replay

When recording a script, Silk Test Workbench recognizes some windows as `Dialog`. If you want to use such a script as a cross-browser script, you have to replace `Dialog` with `Window`, because some browsers do not recognize `Dialog`.

For example, the script might include the following line:
```
/BrowserApplication//Dialog//PushButton[@caption='OK']
```
Rewrite the line to enable cross-browser testing to:
```
/BrowserApplication//Window//PushButton[@caption='OK']
```

DomClick(x, y) Is Not Working Like Click(x, y)

If your application uses the `onclick` event and requires coordinates, the `DomClick` method does not work. Try to use `Click` instead.

FileInputField.DomClick() Will Not Open the Dialog

Try to use `Click` instead.

How Can I Maximize the Browser Window when Starting to Test?

To maximize a browser inside a test script, for example when starting to test, you can use the `Maximize` method of the `BrowserApplication` class.

To maximize your browser, add the following to your test script:

```
'VB code
With _desktop.BrowserApplication("demo_borland_com")

      .Maximize()

End With
```

How can I scroll in a browser?

For VB .NET scripts, Silk Test Workbench provides the following ways to scroll controls in a browser into view during replay:

`ExecuteJavaScript` method (DomElement)	Use the `ScrollIntoView` method to scroll a specific DOM element into the visible area of the browser window.
`ExecuteJavaScript` method (BrowserWindow)	Use the `ExecuteJavaScript` method to scroll the entire page up or down by a specified range.

Examples

The following command scrolls one page down:

```
'VB .NET code
browserWindow.ExecuteJavaScript("window.scrollBy(0,
window.innerHeight)")
```

The following command scrolls down 100 pixels:

```
'VB .NET code
browserWindow.ExecuteJavaScript("window.scrollBy(0, 100)")
```

The following command scrolls up 100 pixels:

```
'VB .NET code
browserWindow.ExecuteJavaScript("window.scrollBy(0, -100)")
```

How Can I See Which Browser I Am Currently Using?

The `BrowserApplication` class provides a property `"browsertype"` that returns the type of the browser. You can add this property to a locator in order to define which browser it matches.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

Examples

To get the browser type, type the following into the locator:

```
browserApplication.GetProperty("browsertype")
```

Additionally, the `BrowserWindow` provides a method `GetUserAgent` that returns the user agent string of the current window.

How do I Verify the Font Type Used for the Text of an Element?

You can access all attributes of the `currentStyle` attribute of a DOM element by separating the attribute name with a `":"`.

Internet Explorer 8 or earlier	`wDomElement.GetProperty("currentStyle:fontName")`

All other browsers, for example Internet Explorer 9 or later and Mozilla Firefox	`wDomElement.GetProperty("currentStyle:font-name")`

I Configured innerText as a Custom Class Attribute, but it Is Not Used in Locators

A maximum length for attributes used in locator strings exists. `InnerText` tends to be lengthy, so it might not be used in the locator. If possible, use `textContents` instead.

I Need Some Functionality that Is Not Exposed by the xBrowser API. What Can I Do?

You can use `ExecuteJavaScript()` to execute JavaScript code directly in your Web application. This way you can build a workaround for nearly everything.

Link.Select Does Not Set the Focus for a Newly Opened Window in Internet Explorer

This is a limitation that can be fixed by changing the Browser Configuration Settings. Set the option to always activate a newly opened window.

Logging Output of My Application Contains Wrong Timestamps

This might be a side effect of the synchronization. To avoid this problem, specify the HTML synchronization mode.

My Test Script Hangs After Navigating to a New Page

This can happen if an AJAX application keeps the browser busy (open connections for Server Push / ActiveX components). Try to set the HTML synchronization mode. Check the *Page Synchronization for xBrowser* topic for other troubleshooting hints.

Recorded an Incorrect Locator

The attributes for the element might change if the mouse hovers over the element. Silk Test Workbench tries to track this scenario, but it fails occasionally. Try to identify the affected attributes and configure Silk Test Workbench to ignore them.

Rectangles Around Elements in Internet Explorer are Misplaced

- Make sure the zoom factor is set to 100%. Otherwise, the rectangles are not placed correctly.
- Ensure that there is no notification bar displayed above the browser window. Silk Test Workbench cannot handle notification bars.

The Move Mouse Setting Is Turned On but All Moves Are Not Recorded. Why Not?

In order to not pollute the script with a lot of useless `MoveMouse` actions, Silk Test Workbench does the following:

- Only records a `MoveMouse` action if the mouse stands still for a specific time.
- Only records `MoveMouse` actions if it observes activity going on after an element was hovered over. In some situations, you might need to add some manual actions to your script.
- Silk Test Workbench supports recording mouse moves only for Web applications, Win32 applications, and Windows Forms applications. Silk Test Workbench does not support recording mouse moves for child technology domains of the xBrowser technology domain, for example Apache Flex and Swing.

What is the Difference Between textContents, innerText, and innerHtml?

- `textContents` is all text contained by an element and all its children that are for formatting purposes only.
- `innerText` returns all text contained by an element and all its child elements.
- `innerHtml` returns all text, including html tags, that is contained by an element.

Consider the following html code.

```
<div id="mylinks">
  This is my <b>link collection</b>:
  <ul>
    <li><a href="www.borland.com">Bye bye <b>Borland</b> </a></li>
    <li><a href="www.microfocus.com">Welcome to <b>Micro Focus</b></a></li>
  </ul>
</div>
```

The following table details the different properties that return.

Code	Returned Value
`browser.DomElement("//div[@id='mylinks']").GetProperty("textContents")`	`This is my link collection:`
`browser.DomElement("//div[@id='mylinks']").GetProperty("innerText")`	`This is my link collection:Bye bye Borland Welcome to Micro Focus`
`browser.DomElement("//div[@id='mylinks']").GetProperty("innerHtml")`	`This is my <b>link collection</b>:<ul> <li><a href="www.borland.com">Bye bye <b>Borland</b></a></li> <li><a href="www.microfocus.com">Welcome to <b>Micro Focus</b></a></li> </ul>`

Note: In Silk Test 13.5 or later, whitespace in texts, which are retrieved through the `textContents` property of an element, is trimmed consistently across all supported browsers. For some browser versions, this whitespace handling differs to Silk Test versions prior to Silk Test 13.5. You can re-enable the old behavior by setting the OPT_COMPATIBILITY option to a version lower than 13.5.0. For example, to set the option to Silk Test 13.0, type the following into your script:

```
'VB .NET code
Agent.SetOption("OPT_COMPATIBILITY", "13.0.0")
```

What Should I Take Care Of When Creating Cross-Browser Scripts?

When you are creating cross-browser scripts, you might encounter one or more of the following issues:

- When recording a script for cross-browser testing, Micro Focus recommends using Google Chrome, Mozilla Firefox, or Microsoft Edge, as a script recorded with Silk Test Workbench against Internet Explorer might slightly differ in comparison to a script recorded on one of the other browsers.
- Different attribute values. For example, colors in Internet Explorer are returned as `"# FF0000"` and in Mozilla Firefox as `"rgb(255,0,0)"`.
- Different attribute names. For example, the font size attribute is called `"fontSize"` in Internet Explorer 8 or earlier and is called `"font-size"` in all other browsers, for example Internet Explorer 9 or later and Mozilla Firefox.

- Some frameworks may render different DOM trees.

Which Locators are Best Suited for Stable Cross Browser Testing?

The built in locator generator attempts to create stable locators. However, it is difficult to generate quality locators if no information is available. In this case, the locator generator uses hierarchical information and indices, which results in fragile locators that are suitable for direct record and replay but ill-suited for stable, daily execution. Furthermore, with cross browser testing, several AJAX frameworks might render different DOM hierarchies for different browsers.

To avoid this issue, use custom IDs for the UI elements of your application.

Why Are the Class and the Style Attributes Not Used in the Locator?

These attributes are on the ignore list because they might change frequently in AJAX applications and therefore result in unstable locators. However, in many situations these attributes can be used to identify objects, so it might make sense to use them in your application.

Why Are Clicks Recorded Differently in Internet Explorer 10?

When you record a `Click` on a `DomElement` in Internet Explorer 10 and the `DomElement` is dismissed after the `Click`, then the recording behavior might not be as expected. If another `DomElement` is located beneath the initial `DomElement`, Silk Test records a `Click`, a `MouseMove`, and a `ReleaseMouse`, instead of recording a single `Click`.

A possible workaround for this unexpected recording behavior depends on the application under test. Usually it is sufficient to delete the unnecessary `MouseMove` and `ReleaseMouse` events from the recorded script.

Why Do I Get an Invalidated-Handle Error?

This topic describes what you can do when Silk Test Workbench displays the following error message: `The handle for this object has been invalidated.`

This message indicates that something caused the object on which you called a method, for example `WaitForProperty`, to disappear. For example, if something causes the browser to navigate to a new page, during a method call in a Web application, all objects on the previous page are automatically invalidated.

When testing a Web application, the reason for this problem might be the built-in synchronization. For example, suppose that the application under test includes a shopping cart, and you have added an item to this shopping cart. You are waiting for the next page to be loaded and for the shopping cart to change its status to `contains items`. If the action, which adds the item, returns too soon, the shopping cart on the first page will be waiting for the status to change while the new page is loaded, causing the shopping cart of the first page to be invalidated. This behavior will result in an invalidated-handle error.

As a workaround, you should wait for an object that is only available on the second page before you verify the status of the shopping cart. As soon as the object is available, you can verify the status of the shopping cart, which is then correctly verified on the second page.

Starting a Browser from a Script

Instead of selecting a browser for replay at the start of a test, you might require to start a specific browser out of the test script during replay.

Using the BrowserBaseState class

Using the `BrowserBaseState` class to start a browser out of a test script ensures that the browser that is specified by the Executable property is running and ready for testing. The base state additionally navigates to the URL that is specified by the Url property and brings the browser to the front.

The following code sample uses the `BrowserBaseState` class to start Internet Explorer.

```
' VB . NET code
Dim MyBrowserBaseState As New BrowserBaseState(BrowserType.InternetExplorer,
"http://www.borland.com")
MyBrowserBaseState.Execute()
```

Using multiple instances of a browser

If you have more then one browser windows or tabs open, Silk Test Workbench handles each browser window or tab as a distinct object with a unique locator. The locators are indexed, for example *WebBrowser* , *WebBrowser[1]* , *WebBrowser[2]* , and so on.

Finding Hidden Input Fields

Hidden input fields are HTML fields for which the tag includes `type="hidden"`. To enable a `Find` to locate hidden input fields, you can use the OPT_XBROWSER_FIND_HIDDEN_INPUT_FIELDS option. The default value of the option is TRUE.

```
'VB .NET code
Agent.SetOption("OPT_XBROWSER_FIND_HIDDEN_INPUT_FIELDS", true)
```

```
// C# code
Agent.SetOption("OPT_XBROWSER_FIND_HIDDEN_INPUT_FIELDS", true);
```

Attributes for Web Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Web applications include:

- caption (supports wildcards *?* and *)
- all DOM attributes (supports wildcards *?* and *)

Note: Empty spaces are handled differently by each browser. As a result, the `textContent` and `innerText` attributes have been normalized. Empty spaces are skipped or replaced by a single space if an empty space is followed by another empty space. Empty spaces are detected spaces, carriage returns, line feeds, and tabs. The matching of such values is normalized also. For example:

```
<a>abc
abc</a>
```

Uses the following locator:

```
//A[@innerText='abc abc']
```

Custom Attributes for Web Applications

HTML defines a common attribute `ID` that can represent a stable identifier. By definition, the ID uniquely identifies an element within a document. Only one element with a specific ID can exist in a document.

However, in many cases, and especially with AJAX applications, the ID is used to dynamically identify the associated server handler for the HTML element, meaning that the ID changes with each creation of the Web document. In such a case the ID is not a stable identifier and is not suitable to identify UI controls in a Web application.

A better alternative for Web applications is to introduce a new custom HTML attribute that is exclusively used to expose UI control information to Silk Test Workbench.

Custom HTML attributes are ignored by browsers and by that do not change the behavior of the AUT. They are accessible through the DOM of the browser. Silk Test Workbench allows you to configure the attribute

that you want to use as the default attribute for identification, even if the attribute is a custom attribute of the control class. To set the custom attribute as the default identification attribute for a specific technology domain, click **Tools** > **Options** > **Record** and select the technology domain.

The application developer just needs to add the additional HTML attribute to the Web element.

Original HTML code:

```
<A HREF="http://abc.com/control=4543772788784322..." <IMG
src="http://abc.com/xxx.gif" width=16 height=16> </A>
```

HTML code with the new custom HTML attribute *AUTOMATION_ID*:

```
<A HREF="http://abc.com/control=4543772788784322..."
AUTOMATION_ID = "AID_Login" <IMG src="http://abc.com/xxx.gif"
width=16 height=16> </A>
```

When configuring the custom attributes, Silk Test Workbench uses the custom attribute to construct a unique locator whenever possible. Web locators look like the following:

```
…//DomLink[@AUTOMATION_ID='AID_Login'
```

Example: Changing ID

One example of a changing ID is the Google Widget Toolkit (GWT), where the ID often holds a dynamic value which changes with every creation of the Web document:

```
ID = 'gwt-uid-<nnn>'
```

In this case <nnn> changes frequently.

xBrowser Class Reference

When you configure an xBrowser application, Silk Test Workbench automatically provides built-in support for testing standard xBrowser controls.

Limitations for Testing on Microsoft Windows 10

The following list lists the known limitations for testing on Microsoft Windows 10 (Windows 10):

- Silk Test does not support testing Universal Windows Platform (UWP) apps on Windows 10.

64-bit Application Support

Silk Test Workbench supports testing 64-bit applications for the following technology types:

- Windows Forms
- Windows Presentation Foundation (WPF)
- Microsoft Windows API-based
- Java AWT/Swing
- Java SWT

Check the *Release Notes* for the most up-to-date information about supported versions, any known issues, and workarounds.

Supported Attribute Types

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests. If necessary, you can change the attribute type in one of the following ways:

- Manually typing another attribute type and value.
- Specifying another preference for the default attribute type by changing the **Preferred attribute list** values.

Attributes for Apache Flex Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Flex applications include:

- automationName
- caption (similar to automationName)
- automationClassName (e.g. `FlexButton`)
- className (the full qualified name of the implementation class, e.g. `mx.controls.Button`)
- automationIndex (the index of the control in the view of the FlexAutomation, e.g. `index:1`)
- index (similar to automationIndex but without the prefix, e.g. `1`)
- id (the id of the control)
- windowId (similar to id)
- label (the label of the control)
- All dynamic locator attributes

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards `?` and `*`.

For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

Attributes for Java AWT/Swing Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Java AWT/Swing include:

- caption
- priorlabel: Helps to identify text input fields by the text of its adjacent label field. Every input field of a form usually has a label that explains the purpose of the input. For controls that do not have a caption, the attribute **priorlabel** is automatically used in the locator. For the **priorlabel** value of a control, for example a text input field, the caption of the closest label at the left side or above the control is used.
- name
- accessibleName
- *Swing only*: All custom object definition attributes set in the widget with `putClientProperty("propertyName", "propertyValue")`

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Attributes for Java SWT Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Java SWT include:

- caption
- all custom object definition attributes

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Attributes for SAP Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for SAP include:

- automationId
- caption

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Locator Attributes for Identifying Silverlight Controls

Supported locator attributes for Silverlight controls include:

- *automationId*
- *caption*
- *className*
- *name*
- All dynamic locator attributes

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

To identify components within Silverlight scripts, you can specify the *automationId*, *caption*, *className*, *name* or any dynamic locator attribute. The *automationId* can be set by the application developer. For example, a locator with an *automationId* might look like `//SLButton[@automationId="okButton"]`.

We recommend using the *automationId* because it is typically the most useful and stable attribute.

Attribute Type	Description	Example
automationId	An identifier that is provided by the developer of the application under test. The Visual Studio designer automatically assigns an *automationId* to every control that is created with the designer. The application developer uses this ID to identify the control in the application code.	`//SLButton[@automationId="okButton"]`
caption	The text that the control displays. When testing a localized application in multiple languages, use the *automationId* or *name* attribute instead of the *caption*.	`//SLButton[@caption="Ok"]`
className	The simple .NET class name (without namespace) of the Silverlight control. Using the *className* attribute can help to identify a custom control that is derived from a standard Silverlight control that Silk Test Workbench recognizes.	`//SLButton[@className='MyCustomButton']`
name	The name of a control. Can be provided by the developer of the application under test.	`//SLButton[@name="okButton"]`

Attention: The *name* attribute in XAML code maps to the locator attribute *automationId*, not to the locator attribute *name*.

During recording, Silk Test Workbench creates a locator for a Silverlight control by using the *automationId*, *name*, *caption*, or *className* attributes in the order that they are listed in the preceding table. For example, if a control has an *automationId* and a *name*, Silk Test Workbench uses the *automationId*, if it is unique, when creating the locator.

The following table shows how an application developer can define a Silverlight button with the text "Ok" in the XAML code of the application:

XAML Code for the Object	Locator to Find the Object from Silk Test
`<Button>Ok</Button>`	`//SLButton[@caption="Ok"]`
`<Button Name="okButton">Ok</Button>`	`//SLButton[@automationId="okButton"]`
`<Button AutomationProperties.AutomationId="okButton">Ok</Button>`	`//SLButton[@automationId="okButton"]`
`<Button AutomationProperties.Name="okButton">Ok</Button>`	`//SLButton[@name="okButton"]`

Locator Attributes for Identifying Rumba Controls

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests. Supported attributes include:

caption The text that the control displays.

priorlabel Since input fields on a form normally have a label explaining the purpose of the input, the intention of **priorlabel** is to identify the text input field, **RumbaTextField**, by the text of its adjacent label field, **RumbaLabel**. If no preceding label is found in the same line of the text field, or if the label at the right side is closer to the text field than the left one, a label on the right side of the text field is used.

StartRow This attribute is not recorded, but you can manually add it to the locator. Use **StartRow** to identify the text input field, **RumbaTextField**, that starts at this row.

StartColumn	This attribute is not recorded, but you can manually add it to the locator. Use **StartColumn** to identify the text input field, **RumbaTextField**, that starts at this column.
All dynamic locator attributes.	For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Attributes for Web Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Web applications include:

- caption (supports wildcards *?* and *)
- all DOM attributes (supports wildcards *?* and *)

Note: Empty spaces are handled differently by each browser. As a result, the `textContent` and `innerText` attributes have been normalized. Empty spaces are skipped or replaced by a single space if an empty space is followed by another empty space. Empty spaces are detected spaces, carriage returns, line feeds, and tabs. The matching of such values is normalized also. For example:

```
<a>abc
abc</a>
```

Uses the following locator:

```
//A[@innerText='abc abc']
```

Attributes for Windows Forms Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Windows Forms applications include:

- automationid
- caption
- windowid
- priorlabel (For controls that do not have a caption, the priorlabel is used as the caption automatically. For controls with a caption, it may be easier to use the caption.)

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Attributes for Windows Presentation Foundation (WPF) Applications

Supported attributes for WPF applications include:

- *automationId*

- *caption*
- *className*
- *name*
- All dynamic locator attributes.

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

Object Recognition

To identify components within WPF scripts, you can specify the *automationId*, *caption*, *className*, or *name*. The name that is given to an element in the application is used as the *automationId* attribute for the locator if available. As a result, most objects can be uniquely identified using only this attribute. For example, a locator with an *automationId* might look like: `//WPFButton[@automationId='okButton']`.

If you define an *automationId* and any other attribute, only the *automationId* is used during replay. If there is no *automationId* defined, the *name* is used to resolve the component. If neither a *name* nor an *automationId* are defined, the *caption* value is used. If no caption is defined, the *className* is used. We recommend using the *automationId* because it is the most useful property.

Attribute Type	Description	Example
automationId	An ID that was provided by the developer of the test application.	`//WPFButton[@automationId='okButton']`
name	The name of a control. The Visual Studio designer automatically assigns a name to every control that is created with the designer. The application developer uses this name to identify the control in the application code.	`//WPFButton[@name='okButton']`
caption	The text that the control displays. When testing a localized application in multiple languages, use the automationId or name attribute instead of the caption.	`//WPFButton[@automationId='Ok']`
className	The simple .NET class name (without namespace) of the WPF control. Using the class name attribute can help to identify a custom control that is derived from a standard WPF control that Silk Test Workbench recognizes.	`//WPFButton[@className='MyCustomButton']`

During recording, Silk Test Workbench creates a locator for a WPF control by using the *automationId*, *name*, *caption*, or *className* attributes in the order that they are listed in the preceding table. For example,

if a control has a *automationId* and a *name*, Silk Test Workbench uses the *automationId* when creating the locator.

The following example shows how an application developer can define a *name* and an *automationId* for a WPF button in the XAML code of the application:

```
<Button Name="okButton" AutomationProperties.AutomationId="okButton"
Click="okButton_Click">Ok</Button>
```

Attributes for Windows API-based Client/Server Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Windows API-based client/server applications include:

- caption
- windowid
- priorlabel: Helps to identify text input fields by the text of its adjacent label field. Every input field of a form usually has a label that explains the purpose of the input. For controls that do not have a caption, the attribute **priorlabel** is automatically used in the locator. For the **priorlabel** value of a control, for example a text box, the caption of the closest label at the left side or above the control is used.

Note: Attribute names are case sensitive, except for mobile applications, where the attribute names are case insensitive. Attribute values are by default case insensitive, but you can change the default setting like any other option. The locator attributes support the wildcards ? and *.

Dynamic Locator Attributes

In a locator for identifying a control during replay you can use a pre-defined set of locator attributes, for example *caption* and *automationId*, which depend on the technology domain. But you can also use every property, including dynamic properties, of a control as locator attribute. A list of available properties for a certain control can be retrieved with the `GetPropertyList` method. All returned properties can be used for identifying a control with a locator.

Note: You can use the `GetProperty` method to retrieve the actual value for a certain property of interest. You can then use this value in your locator.

Example

If you want to identify the button that has the user input focus in a Silverlight application, you can type:

```
browser.Find("//SLButton[@IsKeyboardFocused=true]")
```

or alternatively

```
Dim button = dialog.SLButton("@IsKeyboardFocused=true")
```

This works because Silk Test Workbench exposes a property called `IsDefault` for the Silverlight button control.

Example

If you want to identify a button in a Silverlight application with the font size 12 you can type:

```
Dim button = browser.Find("//SLButton[@FontSize=12]")
```

or alternatively

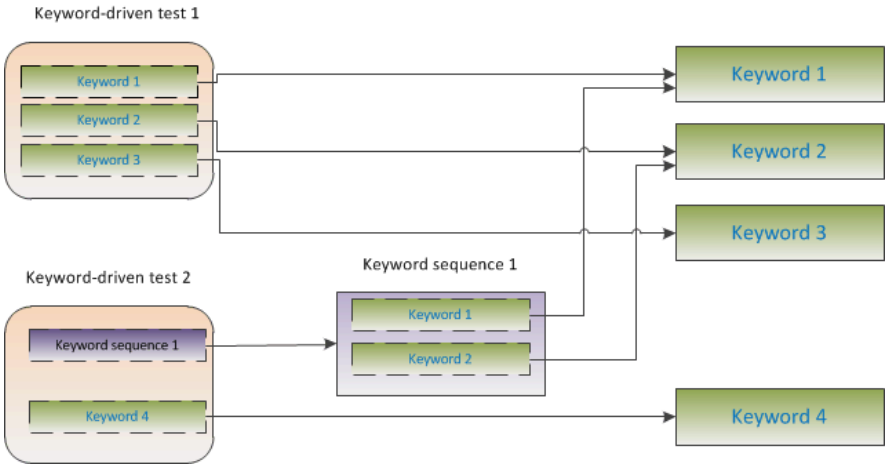
```
Dim button = browser.SLButton("@FontSize=12")
```

This works because the underlying control in the application under test, in this case the Silverlight button, has a property called `FontSize`.

Keyword-Driven Tests

Keyword-driven testing is a software testing methodology that separates test design from test development and therefore allows the involvement of additional professional groups, for example business analysts, in the test automation process. Silk Central and Silk Test support the keyword-driven testing methodology and allow a very close collaboration between automation engineers and business analysts by having automation engineers develop a maintainable automation framework consisting of shared assets in the form of keywords in Silk Test. These keywords can then be used by business analysts either in Silk Test to create new keyword-driven tests or in Silk Central to convert their existing manual test assets to automated tests or to create new keyword-driven tests.

- A *keyword-driven test* is an executable collection of keywords. A keyword-driven test can be played back just like any other test.
- A *keyword sequence* is a keyword that is a combination of other keywords. Keyword sequences bundle often encountered combinations of keywords into a single keyword, enabling you to reduce maintenance effort and to keep your tests well-arranged.
- A *keyword* is a defined combination of one or more actions on a test object. The implementation of a keyword can be done with various tools and programming languages, for example Java or .NET.



There are two phases required to create keyword-driven tests:

1. Designing the test.
2. Implementing the keywords.

For a complete list of the record and replay controls available for keyword-driven testing, see the *Keyword-Driven Testing Class Reference*.

Advantages of Keyword-Driven Testing

The advantages of using the keyword-driven testing methodology are the following:

- Keyword-driven testing separates test automation from test case design, which allows for better division of labor and collaboration between test engineers implementing keywords and subject matter experts designing test cases.
- Tests can be developed early, without requiring access to the application under test, and the keywords can be implemented later.
- Tests can be developed without programming knowledge.

- Keyword-driven tests require less maintenance in the long run. You need to maintain the keywords, and all keyword-driven tests using these keywords are automatically updated.
- Test cases are concise.
- Test cases are easier to read and to understand for a non-technical audience.
- Test cases are easy to modify.
- New test cases can reuse existing keywords, which amongst else makes it easier to achieve a greater test coverage.
- The internal complexity of the keyword implementation is not visible to a user that needs to create or execute a keyword-driven test.

Keywords

A *keyword* is a defined combination of one or more actions on a test object. The implementation of a keyword can be done with various tools and programming languages, for example Java or .NET. In Silk Test Workbench, a keyword is a visual test or a method with the attribute Keyword before the method name. Keywords are saved as keyword assets.

You can define keywords and keyword sequences during the creation of a keyword-driven test and you can then implement them as test methods or visual tests. You can also mark existing test methods or visual tests as keywords with the `Keyword` attribute.Keywords are defined as follows:

```
'VB .NET code
<Keyword("keyword_name")>
```

A *keyword sequence* is a keyword that is a combination of other keywords. Keyword sequences bundle often encountered combinations of keywords into a single keyword, enabling you to reduce maintenance effort and to keep your tests well-arranged.

A keyword or a keyword sequence can have a combined total of 20 input and output parameters. Any parameter of the test method or visual test that implements the keyword is a parameter of the keyword. To specify a different name for a parameter of a keyword, you can use the following:

```
'VB .NET code
Argument("parameter_name")
```

Example

A test method that is marked as a keyword can look like the following:

```
'VB .NET code
<Keyword("Login")>
Public Sub Login()
    ...  // method implementation
End Sub
```

or

```
'VB .NET code
<Keyword("Login", Description:="Logs in with the given name and
password.")>
Public Sub Login(<Argument("UserName")> username As String,
<Argument("Password")> password As String)
    ... // method implementation
End Sub
```

where the keyword logs into the application under test with a given user name and password.

Note: If you are viewing this help topic in PDF format, this code sample might include line-breaks which are not allowed in

scripts. To use this code sample in a script, remove these line-breaks.

- The keyword name parameter of the `Keyword` attribute is optional. You can use the keyword name parameter to specify a different name than the method name. If the parameter is not specified, the name of the method is used as the keyword name.
- The `Argument` attribute is also optional. If a method is marked as a keyword, then all arguments are automatically used as keyword arguments. You can use the `Argument` attribute to specify a different name for the keyword argument, for example *UserName* instead of *userName*.

Creating a Keyword-Driven Test in Silk Test Workbench

Use the **Keyword-Driven Test Editor** to combine new keywords and existing keywords into new keyword-driven tests. New keywords need to be implemented as visual tests or VB .NET methods in a later step.

1. Choose **File** > **New**. The **New Asset** dialog box opens.
2. Select **Keyword-Driven Test** from the asset type list.
3. Type a name for the new asset into the **Asset name** field.
4. Select the project in which the new asset should be included.

   By default, if a project is active, the new asset is created in the active project.

   Note: To optimally use the functionality that Silk Test Workbench provides, create an individual project for each application that you want to test, except when testing multiple applications in the same test.
5. Click **OK**. The **Keyword-Driven Test Editor** opens.
6. Perform one of the following actions:
   - To add a new keyword, type a name for the keyword into the **New Keyword** field.
   - To add an existing keyword, expand the list and select the keyword that you want to add.
7. Press `Enter`.
8. Repeat the previous two steps until the test includes all the keywords that you want to execute.
9. Click **Save**.

Continue with implementing the keywords or with executing the test, if all keywords are already implemented.

Recording a Keyword-Driven Test in Silk Test Workbench

To record a single keyword, see *Recording a Keyword*.

To record a new keyword-driven test:

1. Choose **File** > **New**. The **New Asset** dialog box opens.
2. Select **Keyword-Driven Test** from the asset type list.
3. Type a name for the new asset into the **Asset name** field.
4. Select the project in which the new asset should be included.

   By default, if a project is active, the new asset is created in the active project.

   Note: To optimally use the functionality that Silk Test Workbench provides, create an individual project for each application that you want to test, except when testing multiple applications in the same test.

5. Check the **Begin Recording** check box to start recording immediately.
6. Click **OK**.
7. In the **Record Target** dialog box, perform the following actions:
   a) In the **Asset Type** field, select whether the keyword-driven test should be stored as a .NET script or a visual test.
   b) *Optional:* If you have selected to store the keyword-driven test as a .NET script, specify a name for the .NET script in the **Asset Name** field.
      The default name is the name of the last used .NET script.
   c) Click **OK**.
8. If you have set an application configuration for the current project and you are testing a web application, the **Select Browser** dialog box opens:
   a) Select the browser.
   b) *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
      For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.
   c) *Optional:* Select an **Orientation** for the browser window.
   d) *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.
9. Depending on the dialog that is open, perform one of the following:
   - In the **Select Application** dialog box, click **OK**.
   - In the **Select Browser** dialog box, click **Record**.
10. In the application under test, perform the actions that you want to include in the first keyword.
    For information about the actions available during recording, see *Actions Available During Recording*.
11. To specify a name for the keyword, hover the mouse cursor over the keyword name in the **Recording** window and click **Edit**.

    Note: Silk Test Workbench automatically adds the keyword *Start application* to the start of the keyword-driven test. In this keyword, the applications base state is executed to enable the test to replay correctly. For additional information on the base state, see *Base State*.
12. Type a name for the keyword into the **Keyword name** field.
13. Click **OK**.
14. To record the actions for the next keyword, type a name for the new keyword into the **New keyword name** field and click **Add**. Silk Test Workbench records any new actions into the new keyword.
15. Create new keywords and record the actions for the keywords until you have recorded the entire keyword-driven test.
16. Click **Stop**. The **Record Complete** dialog box opens.

Silk Test Workbench creates the new keyword-driven test with all recorded keywords.

Setting the Base State for a Keyword-Driven Test in Silk Test Workbench

When you execute a keyword-driven test with Silk Test Workbench and the keyword-driven test calls a base state keyword, Silk Test Workbench starts your AUT from the base state.

During the recording of a keyword-driven test, Silk Test Workbench searches in the current project for a base state keyword, which is a keyword for which the `IsBaseState` property is set to *True*.

- If a base state keyword exists in the current project, Silk Test Workbench inserts this keyword as the first keyword of the keyword-driven test.

- If there is no base state keyword in the project, Silk Test Workbench creates a new base state keyword with the name *Start application* and inserts it as the first keyword of the keyword-driven test.

To manually mark a keyword, which is implemented as a .NET method, as a base state keyword, add the `IsBaseState` property to the `Keyword` attribute, and set the value of the property to *True*:

```
<Keyword( "Start application" , IsBaseState:= True )>
Public Sub Start_application()
End Sub
```

To manually mark a keyword, which is implemented as a visual test, as a base state keyword, open the properties of the **<<Start>>** step of the visual test and set the `Is base state` property in the **Keywords** section to **Yes**.

Application configurations for keyword-driven tests are stored in the current project. To manage the application configuration, select **Tools** > **Edit Application Configurations** in the Silk Test Workbench menu. This opens the **Select Project** dialog box with all of the projects to which you have access. Select the project and click **Edit** to edit an application configuration.

During the execution of a keyword-driven test which includes a base state, the application configurations of the project, in which the keyword-driven test is defined, are used.

Implementing a Keyword in Silk Test Workbench

Before implementing a keyword, define the keyword as part of a keyword-driven test.

To implement a keyword for reuse in keyword-driven tests:

1. Open a keyword-driven test that includes the keyword that you want to implement.
2. In the **Keyword-Driven Test Editor**, click **Implement Keyword** to the left of the keyword that you want to implement. The **Implement Keyword** dialog box opens.
3. Select which type of asset the new keyword should be:
   - To implement the keyword as a .NET method, click **.NET Script**. You can add the new keyword to an existing .NET script or create a new .NET script for the keyword by typing a name into the **Asset Name** field.
   - To implement the keyword as a keyword sequence, click **Keyword**. You cannot record a mid-level keyword, as it is a sequence of other keywords.
   - To implement the keyword as a visual test, click **Visual Test**.
4. Type a name for the keyword into the **Asset Name** field.
5. *Optional:* To record the keyword, check the **Start Recording** check box.
6. Click **OK**.
7. Click **Record**.

   For additional information on recording, see *Recording a Keyword*.

Recording a Keyword in Silk Test Workbench

You can only record actions for a keyword that already exists in a keyword-driven test, not for a keyword that is completely new. To record a new keyword-driven test, see *Recording a Keyword-Driven Test*.

To record the actions for a new keyword:

1. Open a keyword-driven test that includes the keyword that you want to record.
2. In the **Keyword-Driven Test Editor**, click **Implement Keyword** to the left of the keyword that you want to implement. The **Implement Keyword** dialog box opens.

3. Select which type of asset the new keyword should be:
   - To implement the keyword as a .NET method, click **.NET Script**. You can add the new keyword to an existing .NET script or create a new .NET script for the keyword by typing a name into the **Asset Name** field.
   - To implement the keyword as a keyword sequence, click **Keyword**. You cannot record a mid-level keyword, as it is a sequence of other keywords.
   - To implement the keyword as a visual test, click **Visual Test**.
4. Check the **Start Recording** check box.
5. Click **OK**.
6. If you have set an application configuration for the current project and you are testing a web application, the **Select Browser** dialog box opens:
   a) Select the browser.
   b) *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
   For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.
   c) *Optional:* Select an **Orientation** for the browser window.
   d) *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.
7. In the application under test, perform the actions that you want to test.
   For information about the actions available during recording, see *Actions Available During Recording*.
8. Click **Stop**. The **Record Complete** dialog box opens.

The recorded actions are displayed in the defined visual test or .NET script.

Marking a Visual Test as a Keyword

You can mark an existing visual test as a keyword which you can then use in keyword-driven tests.

1. Open the visual test that you want to mark as a keyword.
2. Click the **<<Start>>** step in the **Task** pane.
3. Open the **Properties** pane.
4. Set **Is keyword** to **Yes**.

You can now use the visual test as a keyword in a keyword-driven test.

Marking a Test Method in a Script as a Keyword

Mark an existing test method in a VB .NET script as a keyword to reuse the method in keyword-driven tests.

1. Open the VB .NET script which includes the test method that you want to mark as a keyword.
2. Add `<Keyword()>` to the start of the test method.
   By default, the keyword name is the name of the test method.
3. *Optional:* You can set a different name for the keyword by adding `<Keyword("KeywordName")>` to the start of the test method.

You can now use the test method as a keyword in a keyword-driven test.

Examples

To mark the test method `Login` as a new keyword with the name *Login*, type the following before the start of the test method:

```
'VB .NET code
<Keyword("Login")>
```

To mark the test method `Login` as a new keyword with the name *Login* and with the two input parameters `UserName` and `PassWord`, type the following:

```
// VB .NET code
<Keyword("Login", Description="Logs in with the given name and
password.")>
Public Sub Login(Argument("UserName") As String,
Argument("PassWord") As String)
    ...    //
End Sub
```

Note: If you are viewing this help topic in PDF format, this code sample might include line-breaks which are not allowed in scripts. To use this code sample in a script, remove these line-breaks.

Passing Data Between Keywords

You can pass parameters between keywords and you can pass back values from a keyword, which is implemented as a visual test or a .NET script. You cannot pass back the value of visual test global variables to a keyword, as these variables are managed by the visual test execution engine and are not visible to .NET scripts and keywords.

1. If you want to pass data from one keyword to another keyword:
   a) Add an output parameter to the keyword from which you want to pass data.
   b) Add the same parameter as an input parameter to the keyword to which you want to pass data.
2. If you want to pass back data from a keyword, which is implemented as a visual test or a .NET script:
   a) Add an output parameter to the keyword from which you want to pass back data.
   b) In the visual test or the .NET script, to which you want to pass back the data, create a local variable to store the data that is passed in.
   c) Use the local variable within the visual test or .NET script.

Editing a Keyword-Driven Test

Note: In Silk Test Workbench, you can edit and execute keyword-driven tests that are located in Silk Test Workbench, and you can execute keyword-driven tests that are stored in Silk Central. To edit a keyword-driven test, which is stored in Silk Central, open the keyword-driven test in the **Keyword-Driven Test Editor** and click **Edit**.

To edit a keyword-driven test:

1. Open the keyword-driven test in the **Keyword-Driven Test Editor**.
   a) Click **Window** > **Asset Browser**. The **Asset Browser** opens.
   b) Click **Keyword-Driven Test**.
   c) Double-click the keyword-driven test that you want to edit.
2. To add a new keyword to the keyword-driven test:

a) Click into the **New Keyword** field.
b) Type a name for the new keyword.
c) Press `Enter`.

3. To edit an existing keyword, click **Open Keyword** to the left of the keyword.

   Note: Silk Central has the ownership of any keyword that has been created in Silk Central, which means any changes that you make to such keywords are saved in Silk Central, not in Silk Test Workbench.

4. To copy a keyword into the keyword-driven test:
   a) Select the keyword.

      Tip: Use **Ctrl+Click** or **Shift+Click** on the row number column to select multiple keywords.

   b) Press **Ctrl+C**.
   c) Select the row above which you want to insert the keyword.
   d) Press **Ctrl+V**.
5. To move a keyword to another location in the keyword-driven test, click on the keyword and drag it to the new location, or:
   a) Select the keyword.

      Tip: Use **Ctrl+Click** or **Shift+Click** on the row number column to select multiple keywords.

   b) Press **Ctrl+X**.
   c) Select the row above which you want to insert the keyword.
   d) Press **Ctrl+V**.
6. To remove the keyword from the keyword-driven test, click **Delete Keyword** to the left of the keyword.

   The keyword is still available in the **Keywords** window and you can re-add it to the keyword-driven test at any time.
7. To save your changes, click **Save as new version**.

Managing Keywords in a Test in Silk Central

The **Keywords** page enables you to manage the keywords of the selected keyword-driven test. The following actions are possible:

Task	Steps
Opening a test or keyword sequence in Silk Test	Click **Open with Silk Test** to open the selected test or keyword sequence in Silk Test.
Adding a keyword	1. Click **New Keyword** at the bottom of the keywords list, or right-click a keyword and select **Insert Keyword Above** from the context menu. **Note:** You can let Silk Test recommend keywords based on their usage. Toggle the recommendations on or off with **Enable Recommendations** or **Disable Recommendations** in the context menu. For additional information, see *Which Keywords Does Silk Test Workbench Recommend?*. 2. Select a keyword from the list of available keywords or type a new name to create a new keyword. 3. Click **Save**. Alternatively, double click an existing keyword in the **All Keywords** pane on the right or drag and drop it.

Task	Steps
	Tip: You can select multiple keywords with **Ctrl+Click**. When dropping them, they will be sorted in the order that you selected them in.
Deleting a keyword	Click in the **Actions** column of the keyword that you want to delete. Click **Save**.
Changing the order of keywords	Drag and drop a keyword to the desired position. Click **Save**.
Creating a keyword sequence (a keyword consisting of other keywords)	1. Select the keywords that you want to combine in the keywords list. Use **Ctrl +Click** or **Shift+Click** on the row number column to select multiple keywords. 2. Right-click your selection and click **Combine**. 3. Specify a **Name** and **Description** for the new keyword sequence.
Extracting keywords from a keyword sequence	Right-click a keyword sequence and click **Extract keywords**. The original keyword sequence is then replaced by the keywords that it contained, but it is not removed from the library. Click **Save**.
Copying and pasting keywords into tests or keyword sequences	1. Select the keywords that you want to copy in the keywords list. Use **Ctrl +Click** or **Shift+Click** on the row number column to select multiple keywords. 2. Press **Ctrl+C** to copy your selection, or **Ctrl+X** if you want to move the keywords. 3. Open the test or keyword sequence that you want to copy the keywords to and select the row above which the keywords will be inserted. 4. Press **Ctrl+V**. **Tip:** You can also paste your selected keywords into Excel, and copy and paste them from there into your tests or keyword sequences.
Defining parameters for a keyword sequence	1. Click **Parameters** above the keywords list. The **Parameters** dialog box appears. 2. Click **Add Parameter**. 3. Specify a **Name** for the new parameter. If the parameter is an outgoing parameter (delivers a value, instead of requiring an input value), check the **Output** checkbox. 4. Click **OK**. 5. Click **Save**. **Note:** A keyword or a keyword sequence can have a combined total of 20 input and output parameters.
Editing a draft keyword	1. Click in the **Actions** column of the draft keyword that you want to edit. 2. Select a **Group** or specify a new group for the keyword. 3. Type a **Description** for the keyword. This information is valuable for the engineer who will implement the keyword. 4. Click **OK**. 5. *Optional:* Click into a parameter field to add parameters for the keyword. If the keyword is implemented with Silk Test, these parameters will appear in the generated code stub. 6. Click **Save**.

Task	Steps
Searching for a keyword	Use the search field in the **Keywords** window to find a specific keyword. When you enter alphanumeric characters, the list is dynamically updated with all existing matches. Tips for searching: • The search is case-insensitive: `doAction` will find `doaction` and `DOAction`. • Enter only capital letters to perform a so-called *CamelCase* search: `ECD` will find `Enter Car Details`, `Enter Contact Details` and `EnterContactDetails`. • Keyword and group names are considered: `test` will find all keywords that contain `test` and all keywords in groups where the group name contains `test`. • `?` replaces 0-1 characters: `user?test` will find `userTest` and `usersTest`. • `*` replaces 0-n characters: `my*keyword` will find `myKeyword`, `myNewKeyword` and `my_other_keyword`. • `<string>.` only searches in group names: `group.` will find all keywords in groups where the group name contains `group`. • `.<string>` only searches in keyword names: `.keyword` will find all keywords that contain `keyword`. • `<string>.<string>` searches for a keyword in a specific group: `group.word` will find `myKeyword` in the group `myGroup`. • Use quotes to search for an exact match: `'Keyword'` will find `Keyword` and `MyKeyword`, but not `keyword`.

Which Keywords Does Silk Test Workbench Recommend?

When you add keywords to a keyword-driven test or a keyword sequence in the **Keyword-Driven Test Editor**, Silk Test Workbench recommends existing keywords which you might want to use as the next keyword in your test. The recommended keywords are listed on top of the keywords list, and are indicated by a bar graph, with the filled-out portion of the graph corresponding to how much Silk Test Workbench recommends the keyword.

Silk Test Workbench recommends the keywords based on the following:

- When you add the first keyword to a keyword-driven test or a keyword sequence, Silk Test Workbench searches for similar keywords that are used as the first keyword in other keyword-driven tests or keyword sequences. The keywords that are used most frequently are recommended higher.
- When you add additional keywords to a keyword-driven test or a keyword sequence, which already includes other keywords, Silk Test Workbench recommends keywords as follows:
  - If there are keywords before the position in the keyword-driven test or the keyword sequence, to which you add a new keyword, Silk Test Workbench compares the preceding keywords with keyword combinations in all other keyword-driven tests and keyword sequences and recommends the keywords that most frequently follow the preceding combination of keywords.
  - If there are no keywords before the position in the keyword-driven test or the keyword sequence, but there are keywords after the current position, then Silk Test Workbench compares the succeeding keywords with keyword combinations in all other keyword-driven tests and keyword sequences and recommends the keywords that most frequently precede the succeeding combination of keywords.
- Additionally, Silk Test Workbench takes into account how similar the found keywords are. For example, if both the name and group of two keywords match, then Silk Test Workbench recommends these keywords higher in comparison to two keywords for which only the name matches.

- If you have established a connection with Silk Central, any keywords included in keyword-driven tests, which belong to the keyword library that corresponds to the current project, are also considered.

Using Parameters with Keywords

A keyword or a keyword sequence can have a combined total of 20 input and output parameters. This topic describes how you can handle these parameters with Silk Test Workbench, when you are working with keywords that are implemented as VB .NET scripts.

In the **Keyword-Driven Test Editor**, you can view any defined parameters for a keyword or a keyword sequence and you can edit the parameter values.

In the **Keywords** window, you can see which parameters are assigned to a keyword or a keyword sequence when you hover the mouse cursor over the keyword or keyword sequence.

Note: For keywords that are implemented as visual tests, you can specify input and output parameters in the **Input parameters** and **Output parameters** sections in the **Properties** window of the **<<Start>>** step.

Input parameters for simple keywords

You can define and use input parameters for keywords that are implemented as VB .NET scripts in the same way as for any other test method.

The following code sample shows how you can define the keyword `SetUserDetails` with the two input parameters `userName` and `password`:

```
' VB .NET code
<Keyword>
public Sub SetUserDetails(userName As String, password As String)
  ...
End Sub
```

Output parameters for simple keywords

You can define a return value or one or more output parameters for a keyword that is implemented as a VB .NET script. You can also use a combination of a return value and one ore more output parameters.

The following code sample shows how you can define the keyword `GetText` that returns a string:

```
' VB .NET code
<Keyword>
public Function GetText As String
  Return "text"
End Function
```

The following code sample shows how you can define the keyword `GetUserDetails` with the two output parameters `userName` and `password`:

```
' VB .NET code
<Keyword>
public Sub GetUserDetails(ByRef userName As String, ByRef password As String)
  userName="name"
  password="password"
End Sub
```

Parameters for keyword sequences

You can define or edit the parameters for a keyword sequence in the **Parameters** dialog box, which you can open if you click **Parameters** in the **Keyword Sequence Editor**.

Example: Keywords with Parameters

This topic provides an example of how you can use keywords with parameters. A keyword or a keyword sequence can have a combined total of 20 input and output parameters.

As a first step, create a keyword-driven test which contains the keywords that you want to use. You can do this by recording an entire keyword-driven test, or by creating a new keyword-driven test and by adding the keywords in the keyword-driven test editor.

In this example, the keyword-driven test includes the following keywords:

Start application	This is the standard keyword that starts the AUT and sets the base state.
Login	This keyword logs into the AUT with a specific user, identified by a user name and a password.
GetCurrentUser	This keyword returns the name of the user that is currently logged in to the AUT.
AssertEquals	This keyword compares two values.
Logout	This keyword logs the user out from the AUT.

The next step is to add the parameters to the keywords. To do this, open the test scripts of the keywords and add the parameters to the methods.

To add the input parameters `UserName` and `Password` to the keyword `Login`, change

```
' VB .NET code
<Keyword>
public Sub Login()
  ...
End Sub
```

to

```
' VB .NET code
<Keyword>
public Sub Login(UserName As String, Password As String)
  ...
End Sub
```

To add the output parameter `UserName` to the keyword `GetCurrentUser`, change

```
' VB .NET code
<Keyword>
public Sub GetCurrentUser()
  ...
End Sub
```

to

```
' VB .NET code
<Keyword>
public Sub GetCurrentUser(ByRef CurrentUser As String)
  ...
End Sub
```

The keyword-driven test in the **Keyword-Driven Test Editor** should look similar to the following:

		Keyword	Parameters	
1		Start application		
2		Login	*UserName*	*Password*
3		GetCurrentUser	*CurrentUser* ←	
4		AssertEquals	*Expected*	*Actual*
5		Logout		

Now you can specify actual values for the input parameters in the **Keyword-Driven Test Editor**. To retrieve the value of the output parameter `UserName` of the keyword `GetCurrentUser`, provide a variable, for example *${current user}*. You can then pass the value that is stored in the variable to subsequent keywords.

		Keyword	Parameters	
1		Start application		
2		Login	*UserName*	*Password*
3		GetCurrentUser	${current user}	
4		AssertEquals	John Smith	${current user}
5		Logout		

Combining Keywords into Keyword Sequences

Use the **Keyword-Driven Test Editor** to combine keywords, which you want to execute sequentially in multiple keyword-driven tests, into a keyword sequence.

1. Open the keyword-driven test that includes the keywords that you want to combine.
2. In the **Keyword-Driven Test Editor**, press and hold down the `Ctrl` key and then click the keywords that you want to combine.
3. Right-click on the selection and click **Combine**. The **Combine Keywords** dialog box opens.
4. Type a name for the new keyword sequence into the **Name** field.
5. *Optional:* Type a description for the new keyword sequence into the **Description** field.
6. Click **Combine**.

The new keyword sequence opens and is also displayed in the **Keywords** window. You can use the keyword sequence in keyword-driven tests.

Note: Like any other keyword, you cannot execute a keyword sequence on its own, but only as part of a keyword-driven test.

Replaying a Keyword-Driven Test with Specific Variables

Before you can set the values of variables for the execution of a keyword-driven test, you have to create the keyword-driven test.

1. Open the keyword-driven test which you want to execute based on the variables.

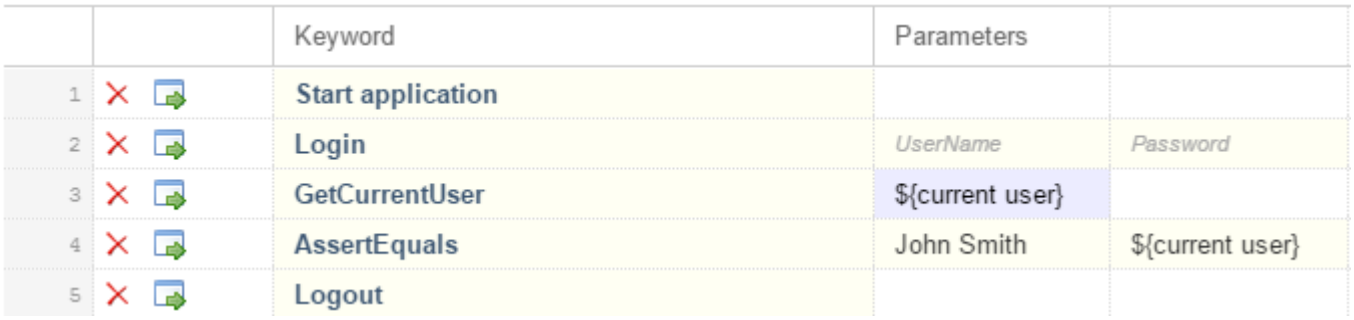
2. In the **Keyword-Driven Test Editor**, click **Global Variables**. The **Global Variables** dialog box opens.
3. In the **Variable Name** column, type the name for the new variable into the first empty field.
4. Type a value for the variable into the **Value** column.

   Note: Do not leave the value field empty, as this might produce wrong test results.
5. Repeat the previous steps until you have specified all variables that should be used for the test execution.
6. Click **OK**.
7. Open the keyword-driven test that you want to execute.
8. In the**Keyword-Driven Test Editor**, edit the parameters to use the new variables.

   Use the following annotation:

   ```
   ${variable name}
   ```

   For example, in the following keyword-driven test, the `${current user}` parameter uses a global variable:



Whenever the keyword-driven test is executed from Silk Test Workbench, the variables are used.

Integrating Silk Test Workbench with Silk Central

Integrate Silk Test Workbench and Silk Central to enable collaboration between technical and less-technical users.

When Silk Test Workbench and Silk Central are integrated and a library with the same name as the active Silk Test Workbench project exists in Silk Central, the **Keywords** view under **View** > **Keywords** displays all keywords from the Silk Central library in addition to any keywords defined in the active Silk Test Workbench project.

Note: The Silk Central connection information is separately stored for every Silk Test Workbench user, which means every Silk Test Workbench user that wants to work with keywords and keyword sequences from Silk Central must integrate Silk Test Workbench with Silk Central.

Integrating Silk Test Workbench with Silk Central provides you with the following advantages:

- Test management and execution is handled by Silk Central.
- Keywords are stored in the Silk Central database (upload library) and are available to all projects in Silk Central.
- Manual tests can be directly automated in Silk Central and the created keyword-driven tests can be executed in Silk Test Workbench from Silk Central.

Note: In Silk Test Workbench, you can edit and execute keyword-driven tests that are located in Silk Test Workbench, and you can execute keyword-driven tests that are stored in Silk Central. To edit a

keyword-driven test, which is stored in Silk Central, open the keyword-driven test in the **Keyword-Driven Test Editor** and click **Edit**.

1. From the menu, select **Tools** > **Options**. The **Options** dialog box opens.
2. In the **Options** tree, select **Silk Central**.
3. Type the URL of your Silk Central server into the **URL** field.

   For example, if the Silk Central server name is *sctm-server*, and the port for Silk Central is 13450, type `http://sctm-server:13450`.
4. Specify the web-service token for authentication.

   You can generate a web-service token in the **User Settings** page of Silk Central, which you can access by clicking on the user name in the Silk Central menu.

   Note: To authenticate with your Silk Central user name and password, you could select **User name and password** from the **Authentication** list. However, for security reasons, Micro Focus recommends using a web-service token for authentication instead of sending your user name and password over the network.
5. Type a valid user name and password into the corresponding fields.
6. Click **Verify** to verify if Silk Test Workbench can access the Silk Central server with the specified user.
7. Click **OK**.

Opening Keywords from Silk Central

To open a keyword in Silk Test Workbench from Silk Central, you can click on the icon to the left of the name of a keyword in Silk Central. This opens the asset that implements the keyword in Silk Test Workbench.

If no instance of Silk Test Workbench is open, you have to enter your username and password or use Windows authentication. Silk Test Workbench creates a DSN and auto-selects it in the **Login** dialog. The database connection information for the database, from which the keyword was originally uploaded to Silk Central, is included.

If an instance of Silk Test Workbench is open, and the instance is connected to the database, from which the keyword was originally uploaded to Silk Central, the keyword is opened in this instance.

Note: If the keyword library was originally uploaded to Silk Central from Silk Test Workbench 16.0, you have to re-upload the library before you open the keyword in Silk Test Workbench. If you do not reload the library, you might experience unexpected behavior when using the keyword.

Implementing Silk Central Keywords in Silk Test Workbench

Before implementing Silk Central keywords, define the keywords as part of a keyword-driven test in Silk Central.

To implement a Silk Central keyword in Silk Test Workbench:

1. Create a project in Silk Test Workbench with the same name as the keyword library in Silk Central, which includes the keyword-driven test.
2. If the keyword library in Silk Central has no type assigned, click **Tools** > **Upload Keyword Library** to set the library type.
3. *Optional:* To implement a specific keyword in Silk Test Workbench from Silk Central, open the **Keywords** tab of the library in Silk Central and click **Implement with Silk Test** in the **Actions** column of the keyword.

4. In Silk Test Workbench, click **View** > **Keywords**.
5. In the **Keywords** window, double-click the keyword-driven test.

   To update the **Keywords** window with any changes from Silk Central, click **Refresh**.
6. In the toolbar, click **Record Actions**.
7. If you have set an application configuration for the current project and you are testing a web application, the **Select Browser** dialog box opens:
   a) Select the browser.
   b) *Optional:* If you want to test a web application on a desktop browser with a predefined browser size, select the browser size from the **Browser size** list.
   For example, to test a web application on Apple Safari and in a browser window which is as big as the screen of the Apple iPhone 7, select **Apple iPhone 7** from the list.
   c) *Optional:* Select an **Orientation** for the browser window.
   d) *Optional:* Click **Edit Browser Sizes** to specify a new browser size and to select which browser sizes should be shown in the **Browser size** list.
8. Click **Record**.

   For additional information on recording, see *Recording a Keyword*.
9. Record the actions for the first unimplemented keyword.
10. When you have recorded all the actions for the current keyword, click **Next Keyword**.
11. To switch between keywords in the **Recording** window, click **Previous Keyword** and **Next Keyword**.
12. Click **Stop**. The **Record Complete** dialog box opens.

Note: You cannot delete keywords or change the sequence of the keywords in a keyword-driven test from Silk Central, as these tests are read only in Silk Test Workbench.

Uploading a Keyword Library to Silk Central

To work with Silk Central, ensure that you have configured a valid Silk Central location. For additional information, see *Integrating Silk Test Workbench with Silk Central*.

To automate manual tests in Silk Central, upload keywords that you have implemented in a Silk Test Workbench project as a keyword library to Silk Central, where you can then use the keywords to automate manual tests.

1. In the menu, select **Tools** > **Upload Keyword Library**. All projects in the current Silk Test Workbench database are listed.
2. Select the project that includes the keywords which you want to upload to Silk Central.

   Note: Ensure that a keyword library with the same name as the project exists in Silk Central.
3. Click **OK**.
4. *Optional:* Provide a description of the changes to the keyword library.
5. *Optional:* Click **Configure** to configure the connection to Silk Central.
6. *Optional:* To see which libraries are available in the connected Silk Central instance, click on the link.
7. Click **Upload**.

   Caution: If the keyword library in Silk Central is already assigned to a different automation tool or another Silk Test client, you are asked if you really want to change the type of the keyword library. Upload the library only if you are sure that you want to change the type.

Silk Test Workbench creates a keyword library out of all the keywords that are implemented in the project. Then Silk Test Workbench saves the keyword library with the name `library.zip` into the output folder of the project. The library is validated for consistency, and any changes which might break existing tests in Silk Central are listed in the **Upload Keyword Library to Silk Central** dialog box. Finally, Silk Test

Workbench uploads the library to Silk Central. You can now use the keywords in Silk Central. Any keyword-driven tests in Silk Central, which use the keywords that are included in the keyword library, automatically use the current implementation of the keywords.

If the current Silk Test Workbench database is an Access database, the uploaded keywords are only available for execution on a machine that has access to that Access database. The keyword library that is uploaded to Silk Central does not include the implementation of the keyword, but just information about which keywords are available in this Silk Test Workbench database.

Uploading a Keyword Library to Silk Central from the Command Line

Upload an external keyword library to Silk Central from a Java-based command line to integrate Silk Central and your keyword-driven tests into your continuous integration build system, for example Jenkins.

To upload your keyword library to Silk Central from a Java-based command line:

1. Select **Help** > **Tools** in Silk Central and download the **Java Keyword Library Tool**.
2. Call the command line tool that is contained in the downloaded `jar` file with the following arguments:
   - `java`
   - `-jar com.borland.silk.keyworddriven.jar`
   - `-upload`
   - `Library name` of the library in Silk Central to be updated, or created if it does not yet exist.
   - `Package name` of the library package (`zip` archive) to be uploaded.
   - `Hostname:port` of the Silk Central front-end server.
   - `Web-service token` of the Silk Central user. Required for authentication. You can generate a web-service token in the **User Settings** page of Silk Central, which you can access by clicking on the user name in the Silk Central menu.

     Note: For security reasons, Micro Focus recommends using a web-service token for authentication instead of sending your user name and password over the network.
   - `Username` of the Silk Central user. Not required when using a web-service token for authentication.
   - `Password` of the Silk Central user. Not required when using a web-service token for authentication.
   - `Update information`, describing the changes that were applied to the library, in quotes.
   - `[-allowUsedKeywordDeletion]`, an optional flag to allow the deletion of keywords that are used in a test or keyword sequence. By default, an error is raised if used keywords are attempted to be deleted.

The following example outlines the command line to upload a library to Silk Central with Java 9 or later:

```
java --add-modules=java.activation,java.xml.ws -jar
com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scLogin
scPassword "Build xy: Implemented missing keywords"
```

Examples

The following example outlines the command line to upload a library to Silk Central with Java 8 or prior by using a web-service token for authentication:

```
java -jar com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scToken
"Build xy: Implemented missing keywords"
```

To upload the same library with Java 8 or prior by using user name and password for authentication, use a command like the following:

```
java -jar com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scLogin
scPassword "Build xy: Implemented missing keywords"
```

The corresponding commands with Java 9 or later are:

```
java --add-modules=java.activation,java.xml.ws -jar
com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scToken
"Build xy: Implemented missing keywords"
```

```
java --add-modules=java.activation,java.xml.ws -jar
com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scLogin
scPassword "Build xy: Implemented missing keywords"
```

Note: When uploading a keyword-driven library with Java 9 or later, ensure `JAVA_HOME` is defined on the execution servers and points to a JDK with the corresponding Java version.

Searching for a Keyword

Use the search field in the **Keywords** window to find a specific keyword. When you enter alphanumeric characters, the list is dynamically updated with all existing matches. Tips for searching:

- The search is case-insensitive: `doAction` will find `doaction` and `DOAction`.
- Enter only capital letters to perform a so-called *CamelCase* search: `ECD` will find `Enter Car Details`, `Enter Contact Details` and `EnterContactDetails`.
- Keyword and group names are considered: `test` will find all keywords that contain `test` and all keywords in groups where the group name contains `test`.
- ? replaces 0-1 characters: `user?test` will find `userTest` and `usersTest`.
- * replaces 0-n characters: `my*keyword` will find `myKeyword`, `myNewKeyword` and `my_other_keyword`.
- `<string>.` only searches in group names: `group.` will find all keywords in groups where the group name contains `group`.
- `.<string>` only searches in keyword names: `.keyword` will find all keywords that contain `keyword`.
- `<string>.<string>` searches for a keyword in a specific group: `group.word` will find `myKeyword` in the group `myGroup`.
- Use quotes to search for an exact match: `'Keyword'` will find `Keyword` and `MyKeyword`, but not `keyword`.

Filtering Keywords

To find a specific keyword in the current project, you can filter the keywords that are displayed in the **Keywords** window. If an integration with Silk Central is configured, the result includes the relevant keywords from Silk Central.

1. In the menu, click **View** > **Keywords**. The **Keywords** window opens.
2. In the **Keywords** window, type the name of the keyword that you are searching for into the search field. The **Keywords** window lists all keywords in the current project with the given name.
3. *Optional:* To edit a keyword, hover the mouse cursor over the keyword in the **Keywords** window and click **Go to implementation**.

Grouping Keywords

To better structure the keywords in a library, you can group them.

This topic shows how you can add a keyword to a specific group These group names are also used by Silk Central and your keywords are grouped accordingly.

To add a keyword to a specific group:

1. Open the implementation of the keyword.
   a) Activate the project in which the keyword is implemented.
   b) Open the **Keywords** window.
   c) In the **Keywords** window, select the keyword.
   d) Click **Go to implementation**.
2. If the keyword is implemented in a VB .NET script, add the keyword group to the start of the script.
   For example, to add the group calculator to the keyword, type:
   ```
   <KeywordGroup("Calculator")>
   ```
3. If the keyword is implemented in a visual test, add the keyword group to the properties of the visual test.
   a) Select the start step of the visual test.
   b) Open the **Properties** pane.
   c) Type the name of the group into the **Keyword group** property.

In the **Keywords** window, the displayed keyword name now includes the group. For example, the keyword *Addition* in the group *Calculator* is displayed as `Calculator.Addition`.

Command Line

Silk Test Workbench contains two command line programs that you can use to open and execute tests.

SilkTest.Exe Command Line

You can use the `SilkTest.exe` command line program to open Silk Test Workbench and bypass the login dialog box. It is located in: `\Silk\Silk Test\ng\gui\`.

Parameters

Parameter Long Name	Parameter Short Name	Description
`userid` or `user`	`-u`	The username that you want to login with.
`pass` or `password`	`-p`	The password of the user for login.
`dsn`	`-d`	The DSN that you want to use.

Example

```
SilkTest.exe -u admin -p mypassword -d mydsn
```

STW.EXE Command Line

Silk Test Workbench includes a command line program that you can use to execute one or more .NET scripts or visual tests for batch script execution. The command line will return error information as well as optionally create an output file for results.

The command line program is named `stw.exe` and is located in: `\Silk\Silk Test\ng\gui\`.

The syntax for the command line is:

```
STW -d DSN_Name -u User_Name -p User_Password -s Script_Name
```

The topics in this section describe how to run the program with the required and optional parameters.

Parameters for STW.EXE

The command line program contains both mandatory and optional parameters. Parameters can be specified using either the long or short form of its name. Some of the optional parameters allow you to do things such as change your environment, send variables with a script, and create an output file.

Note: Parameter names, scripts, and script parameters are not case sensitive. However, the *values* of script parameters are passed through to the script as specified.

Note: You can use the `/` character instead of the `-` character to preface the parameter name.

Parameter Long Name	Parameter Short Name	Required	Description
`dsn`	`-d`	Yes	The name of the DSN to use to connect to the database. Although this parameter is required, if left blank, *SilkTest* will be used.
`username`	`-u`	Yes	The name of the user used to connect. If left blank, *Admin* will be used.
`password`	`-p`	No	The password of the user used to connect. Optional if the password is `Nothing` (this is not recommended).
`project`	`-r`	No	The project that contains the scripts to play back. By default, all scripts in the project will execute. Use the `-script` parameter to specify unique scripts. If you do not specify a script and a script requires input parameters, the default values will be used. .NET scripts are executed first and visual tests are executed second. If the project does not exist, an error message will display.  **Important:** If the project parameter is not specified, then the *Common* project is used by default. For example, if you write `stw -d silktest -u admin -p admin -s Script1 -r OtherProject -s Script2`, *Script1* will be run in the *Common* project, while *Script2* will be run in the project called *OtherProject*.
`environment`	`-e`	No	The name of the run environment for the scripts to use. The default playback environment will be used if this parameter is not included. If the playback environment does not exist, then the script will not be executed.  **Important:** If used, this parameter must precede the parameters `project` and `script`.
`script`	`-s`	Yes	The name of the .NET script, visual test, or keyword-driven test to play back. If the .NET script, visual test, or keyword-driven test does not exist, an error message will display.
`variable`	`-m`	No	This parameter allows you to specify the parameters for a .NET script, a visual test, or a keyword-driven test. For example, you can use this parameter to pass global properties from Silk Central to a keyword-driven test. This overrides any global properties that are defined in the asset. If you use this parameter, it must follow the `script` parameter. The name and value pair should be enclosed in double quotes. Use the = character to denote a value that is assigned to a variable. The Boolean parameter should be specified as `True` or `False.` Parameters can be specified in any order but the names and numbers must match the parameters of the script exactly.

Parameter Long Name	Parameter Short Name	Required	Description
			Passing too many, not enough, or invalid names will cause an error message and the script will not run.
`file`	`-f`	No	This option allows you to specify command line parameters in a text file and feed that to the command line program. This could be used if you have lengthy arguments that you don't want to re-type. See *Input File* for full usage details.
`outputfile`	`-o`	No	Use this parameter to specify the name of a file that will contain information about the scripts that were executed. Only scripts that execute and either complete successfully or generate a playback error will appear in the output file. The name of the output file must appear in quotes if the path or name contains spaces. See *Output File* for full usage details.
`browser`	`-b`	No	Use this parameter to specify on which browser scripts that have browser application configurations are executed. The parameter is case insensitive and the following values are allowed: • `InternetExplorer` • `Firefox` • `GoogleChrome` • `Edge` • `AndroidBrowser` • `Safari` • `None` When you want to use `AndroidBrowser` or `Safari`, specify the `mobiledevice` parameter. **Important:** If used, this parameter must precede the parameters `project` and `script`.
`commandlineargs`	`-cla`	No	Use this parameter to specify command line arguments for the base state of the script. You can use this parameter with any base state that uses command line arguments. If the agent uses the command line arguments, they will be honored. Otherwise they will be ignored. You can also use the *silktest.commandLineArgs* environment variable to specify command line arguments. If command line arguments are specified in both the environment variable and as parameters to `STW.exe`, the parameters to `STW.exe` are used. **Important:** If used, this parameter must precede the parameter `script`.
`viewportwidth`	`-vw`	No	Use this parameter to specify the width of the browser viewport in pixels. You can only use the `viewportwidth` parameter in combination with the `viewportheight` parameter. Specifying only one of the parameters will result in

Parameter Long Name	Parameter Short Name	Required	Description
			an error. You cannot use this parameter in combination with the `viewportname` parameter.
`viewportheight`	`-vh`	No	Use this parameter to specify the height of the browser viewport in pixels. You can only use the `viewportheigth` parameter in combination with the `viewportwidth` parameter. Specifying only one of the parameters will result in an error. You cannot use this parameter in combination with the `viewportname` parameter.
`viewportname`	`-vn`	No	Use this parameter to specify the width and height of the viewport by using the predefined size of a custom browser from the list in the UI. This list is visible when selecting a browser in the **Select Browser** dialog, the **Web** tab of the **Select Application** dialog, and the **Edit Browser Application Configuration** dialog. You cannot use this parameter in combination with the `viewportwidth` and `viewportheigth` parameters.
`orientation`	`-on`	No	The `orientation` parameter is only considered when the `viewportname` is specified. Its values can be *portrait* or *landscape*. This is equivalent to the **Orientation** setting in the UI.
`mobiledevice`	`-md`	No	The name of the mobile device that the script is executed against. This name also displays in the **Select Browser** dialog box. To reset this value, pass in an empty string as two double quotes (""). **Important:** If used, this parameter must precede the parameters `project` and `script`.
`append`	`-a`	No	Use this parameter to append the result. This is the same setting as displayed in the **Options** dialog box.
`increment`	`-i`	No	Use this parameter to increment the result. This is the same setting as displayed in the **Options** dialog box.
`verbose`	`-v`	No	This parameter displays output to the user during the execution of the script. It can appear anywhere in the command line. DSN, username, and password are always displayed. If an output file is used, it will be displayed. Where a script name is shown, the project name will appear following the name in brackets . A line containing the name of the script, project, script status, total verifications, passed verifications, and failed verifications will be shown for each script executed.
`help` or `?`	`-h`	No	Type `STW -?` to display the available parameters for the command line program.
`resultdir`	`-rd`	No	Use this parameter to specify a directory into which all result files of the executed scripts are stored. The following files are stored in this directory: • The XML result files. • The XSLT files that correspond to the result XML files.

Parameter Long Name	Parameter Short Name	Required	Description
			• Error screens, if any errors occurred during the execution.
`connectionstring`	`-c`	No	Use this parameter to test a browser or a mobile device which is connected to a remote location. When executing a test with the `connectionstring` parameter, the `connectionstring` is written to the output file, between the mobile device and the parameters.

Example 1

This example shows the command to execute a browser script on Internet Explorer.

```
STW –dsn silktest –browser InternetExplorer –script MyScript
```

Example 2

This example shows the command to execute a browser script on both Internet Explorer and Google Chrome.

```
STW –dsn silktest –browser InternetExplorer –script MyScript –
browser GoogleChrome –script MyScript
```

Example 3

This example shows the command to execute a script first against the Android browser and then on the browser that is set in the application configuration of the script.

```
STW –dsn silktest –browser AndroidBrowser –mobiledevice Nexus_4
–script MyScript –browser None –mobiledevice "" –script MyScript
```

Example 4

This example shows the command to execute all scripts in the Common project against Chrome for Android on a mobile device.

```
STW –dsn silktest –browser GoogleChrome –mobiledevice Nexus_4 –
project Common
```

Example 5

This example shows the command to execute all scripts in the Common project against Chrome for Android on a mobile device that is connected to a Mac.

```
STW –dsn silktest -connectionstring
"platformName=Android;deviceName=Moto G;host=http://MyMac -
GoogleChrome" –project Common
```

Example 6

This example shows the command to execute a browser script on Internet Explorer with a browser viewport of size 800x600.

```
STW –dsn silktest –browser InternetExplorer -viewportwidth 800 -
vieportheight 600 –script MyScript
```

Example 7

This example shows the command to execute a browser script on Internet Explorer with the browser viewport size SVGA (800, 600).

```
STW –dsn silktest –browser InternetExplorer -viewportname SVGA –
script MyScript
```

Example 8

This example shows the command to execute a browser script on Mozilla Firefox with a custom profile:

```
STW –dsn silktest -username admin -password admin -browser
Firefox -commandLineArgs "-p MyProfile" -project Common -script
BrowserTest
```

Input File

The `-file` option allows you to specify command line parameters in a text file and feed that to the command line program.

The file can be specified in any location along the command line, but it is processed in the order that it appears and may affect command line options before or after it. You can specify multiple `-file` commands (followed by file names) and files can also contain `-f` parameters. Parameters can appear on multiple lines.

Use the # character to comment out a line.

If a file contains another **-file** parameter, recursion usage will be checked. This means that a file cannot have a `-file` parameter for the same file referenced anywhere in the same processing set.

Files can be in the following formats: UTF-8, UTF-16, or Unicode.

Example 1: FileA.txt

```
# Execute my script
-script MyScript
# Change the environment and run again
-environment MyEnvironment
-script MyScript
```

Example 2: FileB.txt

```
# Include contents of FileA.txt, twice
-file FileA.txt

# Reset the playback environment
-environment

# Include it again
-file FileA.txt
```

Output File

When the `outputfile` parameter is used, the command line will attempt to create the specified file and overwrite the contents. If the file cannot be accessed, an error message will display, but execution will proceed. This may occur for the following reasons:

- The path does not exist.
- This is locked or in use by another process.
- The user does not have access rights to the folder.

Output File Format

The file that is created is a text file where each line represents a script that was executed. Each line contains a number of fields separated by a tab character (`0x9`). This is the `TXT` format that is used in Microsoft Office Excel.

Output File Fields

Field	Description
Script name	This is either a .NET or Visual test script.
Project	The project that the script exists within. If no project is specified in the command line, "Common" is shown here.
Run Environment	The name of the run environment that was used to execute the script. This field is empty if the default run environment was used.
Total Verifications	The total number of verifications that were executed by the script.
Verifications Passed	The number of verifications that passed
Verifications Failed	The number of verifications that failed.
Script Status	The status of the script.
Playback Error	Contains the playback error message. This is empty if no playback error is detected.
Browser	The name of the browser that was supplied when the script was executed.
Connection String	The connection string to a mobile device or a remote location.
Script parameters	If the script takes parameters they are listed here in pairs of fields. The name of the parameter in one field and the value in the next.

Errors

The command line program will return an `ERRORLEVEL` to the operating system which can be analyzed by a calling program to determine what happened. The following table describes the errors.

Code	Description	Returned as Script Status in output file
0	No errors. All scripts were executed successfully.	Yes
1	Fatal error. The command line program could not execute at all. One reason could be that a language resource file is missing	No
2	Failed to create the Silk Test COM object. Check that the COM object has been registered. Note that running Silk Test.exe registers the COM object.	No

Code	Description	Returned as Script Status in output file
3	DSN, username, or password incorrect. For additional information, see *Prerequisites to Run STW.EXE with Silk Central*.	No
4	No script name was specified.	No
5	The specified project was not found.	No
6	The specified script was not found.	No
7	The specified run environment was not found.	No
8	Invalid script parameters. Too few or too many script parameters were specified, or a parameter was not found.	No
9	The script did not complete execution. It may have stopped before it reached the end but not a result of a playback error.	Yes
10	The script failed with playback errors.	Yes
11	The script failed to execute or the script contained verifications that failed.	Yes
12	Multiple issues have occurred. For example, a script failed and a script playback error was detected.	No
13	This is returned if the command line could not be processed due to the `-file` option. For example, if the file does not exist, or there is file recursion.	No
14	There are various different ways that the command line window can be stopped or shut down. `Ctrl+C`, `Ctrl+Break`, logging out of Windows, or shutting down Windows will all close or stop the command line. These will be caught and this error code will be returned. Closing the window will also stop the script, log out of the product, and unload `SilkTest.EXE`.	No
15	Only one of the viewportwidth and the viewportheight were specified.	No
16	The viewportname was specified together with the viewportwidth and the viewportheight.	No

Checking for the ERRORLEVEL Using a Batch File

There are a number of ways to check for the `ERRORLEVEL` in a batch file. The following example describes one approach.

```
@ECHO OFF
SETLOCAL

REM
-------------------------------------------------------------------------------
REM Define the error codes that can be returned by STW.EXE
REM
-------------------------------------------------------------------------------
SET eNoError=0
SET eFatalError=2
SET eUnableToCreateComObject=2
SET eLoginFailed=3
SET eNoScriptSpecified=4
SET eProjectNotFound=5
SET eScriptNotFound=6
SET eRunEnvNotFound=7
SET eInvalidScriptParameters=8
SET eScriptDidNotComplete=9
SET eScriptPlaybackError=10
SET eScriptFailed=11
```

```
SET eMultipleProblems=12
SET eCommandCouldNotBeProcessed=13
SET ePrograTerminatedByUser=14
SET eInvalidViewportSize=15
SET eInvalidViewportParameters=16

REM
-------------------------------------------------------------------------------
REM Run STW.EXE
REM
-------------------------------------------------------------------------------
@ECHO ON
STW.EXE -username Admin -dsn STW-Scratch -script ScriptNotFound
@ECHO.
@ECHO OFF

REM
-------------------------------------------------------------------------------
REM Anything greater than 12 is unknown, can be caused if the command
REM processor cannot find STW.EXE.
REM
-------------------------------------------------------------------------------
IF %ERRORLEVEL% GTR %eMultipleProblems% (
    ECHO STW.EXE returned an unknown return code %ERRORLEVEL%
    GOTO END
)

REM
-------------------------------------------------------------------------------
REM Check the specific error codes here.
REM
-------------------------------------------------------------------------------
IF %ERRORLEVEL% EQU %eMultipleProblems% (
    ECHO eMultipleProblems
    GOTO END
)

IF %ERRORLEVEL% EQU %eScriptFailed% (
    ECHO eScriptFailed
    GOTO END
)

IF %ERRORLEVEL% EQU %eScriptPlaybackError% (
    ECHO eScriptPlaybackError
    GOTO END
)

IF %ERRORLEVEL% EQU %eScriptDidNotComplete% (
    ECHO eScriptDidNotComplete
    GOTO END
)

IF %ERRORLEVEL% EQU %eInvalidScriptParameters% (
    ECHO eInvalidScriptParameters
    GOTO END
)

IF %ERRORLEVEL% EQU %eRunEnvNotFound% (
    ECHO eRunEnvNotFound
    GOTO END
)

IF %ERRORLEVEL% EQU %eScriptNotFound% (
    ECHO eScriptNotFound
```

```
    GOTO END
)

IF %ERRORLEVEL% EQU %eProjectNotFound% (
    ECHO eProjectNotFound
    GOTO END
)

IF %ERRORLEVEL% EQU %eNoScriptSpecified% (
    ECHO eNoScriptSpecified
    GOTO END
)

IF %ERRORLEVEL% EQU %eLoginFailed% (
    ECHO eLoginFailed
    GOTO END
)

IF %ERRORLEVEL% EQU %eUnableToCreateComObject% (
    ECHO eUnableToCreateComObject
    GOTO END
)

IF %ERRORLEVEL% EQU %eFatalError% (
    ECHO eFatalError
    GOTO END
)

IF %ERRORLEVEL% EQU %eNoError% (
    ECHO eNoError
    GOTO END
)

IF %ERRORLEVEL% EQU %eCommandCouldNotBeProcessed% (
    ECHO eCommandCouldNotBeProcessed
    GOTO END

IF %ERRORLEVEL% EQU %ePrograTerminatedByUser% (
    ECHO ePrograTerminatedByUser
    GOTO END
)

IF %ERRORLEVEL% EQU %eInvalidViewportSize% (
    ECHO eInvalidViewportSize
    GOTO END
)

IF %ERRORLEVEL% EQU %eInvalidViewportParameters% (
    ECHO eInvalidViewportParameters
    GOTO END
)

:END

ENDLOCAL
```

Prerequisites to Run STW.EXE with Silk Central

To run STW.EXE with a Silk Central execution server and to successfully connect to a Silk Test Workbench database, the following prerequisites are required:

1. Run the Silk Central execution server as a process, and not as a service.

2. Use database authentication to connect to the Silk Test Workbench database. Silk Test Workbench can use Windows authentication or database authentication when connecting to a Silk Test Workbench SQL Server database.
3. Configure the database server instance to use a static port.

If these prerequisites are not met, Silk Test Workbench might return the following error when trying to run STW.EXE with a Silk Central execution server:
Error 3: DSN, username, or password incorrect.

Examples for Using STW.EXE

The command line supports the execution of any number of scripts in one execution of `STW.EXE`. The command line also supports input and output files and changing the project and run environments on a script-by-script basis.

Note: The `-d`, `-u`, and `-p` parameters are not included in most of these examples for readability.

General Help

```
STW -help
```

Single Project, All Scripts

```
STW -project ProjectName
```

Single Project, Single Script

```
STW -project ProjectName -script ScriptName
```

Single Script, Common Project

```
STW -s scriptname
```

The command line will use `SilkTest` as the DSN and `Admin` as the username with no password.

Script with a Space in the Name

```
STW -script "Script with spaces"
```

Specific Scripts in the Common project

```
STW -script Script1 -script Script2
```

All scripts in the Common project

```
STW
```

Specific Scripts in Different Projects

In this example two different scripts are executed, since you can have the same script name in different projects, these refer to two unique scripts.

```
STW -project Project1 -script Script1 -project Project2 -script Script1
```

Specific Scripts in the Common and User Projects

Note: Once a project name is specified, all scripts that follow will be associated with that project.

```
STW -project Project1 -script Script1 -script Script2 -project Common -script
Script1 -script Script2
```

Using an Input File

```
STW –file FileA.txt
```

Where the following is contained within `FileA.txt`:

```
# Execute my script
-script MyScript
# Change the environment and run again
-environment MyEnvironment
-script MyScript
```

Environment

```
STW -environment PlaybackEnvironment -s ScriptName
```

Multple Run environments

This example shows the same script being executed with a user defined run environment and the default run environment.

```
STW -environment MyRunEnv -script Script1 -environment -script Script1
```

The second use of `-environment` does not specify a name, it just resets the environment back to the default.

Variables

When executing a .NET script, visual test, or keyword-driven test that has parameters through the command line, you can specify the parameters of the asset by using the *variable* parameter. This overrides any global properties that are defined in the asset.

```
STW -script ScriptWithParameters -variable "var1=1234" "var2=Hello"
"var3=True"
```

Verbose Mode

```
STW -u Admin -verbose -d MyDsn -s MyScript
```

The following is an example of output in verbose mode:

```
STW.exe
Copyright (c) 1996-2011, Micro Focus IP Development Limited. All Rights
Reserved

DSN                   : STW-Scratch
Username              : admin
Password              :
Script(s)             : MyScript [Project3]

Append result         : False
Increment result      : False
Output file           : MyOutput.txt
Verbose Mode          : ON

Searching executable assets...

.NET Script ' MyScript' [Project3] found

Executing MyScript [Project3] ...

Script: MyScript [Project3], Status: Failed, Verifications: Total 5, Passed
0, Failed 5
```

OutputFile

```
STW -outputfile C:\Temp\MyFile.txt
```

Using a result directory

```
STW -s scriptname -resultdir C:\Temp\TestResults
```

or

```
STW -s scriptname -rd C:\Temp\TestResults
```

Command Line Asset Import and Export

Silk Test Workbench's import utility (`STWImport.exe`) and export utility (`STWExport.exe`) can be executed directly from a command line prompt to import or export assets from your database.

You can also create a reusable configuration file to store regularly used parameter values.

Note: When using Silk Test Workbench 18.5 or later, you cannot import assets that have been exported with a Silk Test Workbench version prior to 18.5. To import such exported assets, use the **StwConvertExportedXML** program. For additional information, see *Importing Assets*.

Asset Import Command Line Parameters

Use the `STWImport.exe` command line utility to import assets from the command line. Import parameters can be specified entirely from a command line, or unless otherwise noted, specified in a configuration file.

The utility is located in `\Silk\Silk Test\ng\gui\`

Command Line Import Parameters

Syntax	Values	Required	Description
`Username (-u)`	User-defined	Yes	The name of the user used to connect. It can only be specified using the command line. Any configuration file specification for this parameter is ignored. When using Windows authentication, omit the user name.
`Password (-p)`	User-defined	Yes	The password of the user used to connect. This parameter is required if the password is not `Nothing`. It can only be specified using the command line. Any configuration file specification for this parameter is ignored. When using Windows authentication, omit the password.
`DSN (-d)`	User-defined	Yes	The name of the DSN to use to connect to the database. This parameter is required for any DSN names other than "SilkTest". It can only be specified using the command line. Any configuration file specification for this parameter is ignored.
`ImportFile (-if)`	User-defined		The file name or location of the ZIP file to be imported.
`Execute (-ex)`	Yes | No (see Note below)	Yes	Commits the import operation. The default value is "No".
`Filename (-f)`	User-defined	No	An INI configuration file that stores command line parameters. It can only be specified using the command line. **Note:** The INI file parameters override the command line parameters with one exception. The Projects (-prj) and System (-s) parameter values specified in the INI file do not override the same parameters specified on the

Syntax	Values	Required	Description
			command line. Instead the values for each are both used.
`AllVersions (-av)`	Yes | No (see Note below)	No	Imports all versions of the asset(s). The default value is "No".
`ReportResult (-rr)`	User-defined	No	Contains the file name and location of the summary report file to be created. The default location is the current directory.
`SilentMode (-sm)`	Yes | No (see Note below)	No	Prevents the display of the console output.
`Projectmapping (-pm)`	User-defined	No	Specifies the target project to use instead of the original source project from the import file. Multiple mappings can be separated by using the pipe character (|).

`Projects (-prj)`	Asset nodes	No	Specifies a single project asset or group of project assets for import. All assets in the database will be imported without this parameter. Groups of nodes are separated by the pipe character(|). The format is: `{ProjectName}.{AssetType}.{AssetName}.{Ver#}`. For example:
			`-prj ProjectA.Script.SampleScript.1`
			Strings separated by the pipe character must be included within double quotes. For example:
			`-prj "Common | ProjectB.Result"`
			Note: When specifying the asset type, use the singular form of the asset. For example, use "result" not "results". For asset names comprised of two words (visual test), remove the space (visualtest). There is one exception: test script. Use only the word "script" to specify a test script.

`System (-s)`	Asset nodes	No	Specifies a single system asset or group of system assets for import. All system assets in the database will be imported without this parameter. Groups of nodes are separated by the pipe character(|). The format is: `{SystemName}.{SystemType}.{SystemAssetName}`. For example: `-s Options.Record.RecA` Strings separated by the pipe character must be included within double quotes.
`ProjectAssetResolve- Conflict (-parc)`	`s|a|p`	No	Specifies the conflict resolution method for project assets that already exist in the database. • s stops the import. • a adds a version to the asset upon import. • p prompts the user to choose an action for each conflicting asset and overrides the SilentMode parameter. p is the default.

Syntax	Values	Required	Description
`SystemAssetResolve- Conflict (-sarc)`	`s|p`	No	Specifies the conflict resolution method for system assets that already exist in the database. • s stops the import. • p prompts the user to choose an action for each conflicting asset and overrides the SilentMode parameter. 'p' is the default.

Note: To specify the default value of a command line parameter, simply omit the parameter from the command. To specify the non-default value, include only the parameter without its value. For example, to import all versions of an asset, include the `AllVersions (-av)` parameter without a "Yes" value from your import command. To import only the latest version of an asset, simply omit this parameter to effect the default value of "No".

When specifying parameter values in the import configuration file, the default value is used whenever a parameter or parameter value is omitted. For non-default values, the value must be explicitly stated. For example, to import all versions of an asset, include the AllVersions (-av) parameter together with the value "Yes".

Note: When using Silk Test Workbench 18.5 or later, you cannot import assets that have been exported with a Silk Test Workbench version prior to 18.5. To import such exported assets, use the **StwConvertExportedXML** program. For additional information, see *Importing Assets*.

Example

```
C:/>STWImport.exe -u Admin -p admin -d MyDBName -mt sf -mn "c:
\MySTAssetFiles" -prj ProjectA -pm "ProjectA=ProjectB" -rr "c:\STImport.txt" -
ex
```

In the example above, the latest version of ProjectA asset files contained in the MySTAssetFiles directory are imported into ProjectB in the MYDBName database and a file containing a summary report is created.

Example with Configuration File

```
C:/>STWImport.exe -u Admin -p admin -d MyDBName -f "C:\STimport.ini"
```

In the example above, only the import parameters that are required to be specified on the command line are included. All other parameters are stored in the configuration file.

Creating a Command Line Import Configuration File

To permanently store settings, you can create a command line import configuration file. Most import parameters can be stored in this file. Several parameters such as `Username`, `Password`, `DSN`, and `Filename` must be specified on the command line.

1. Create an `.INI` file to hold all of the optional import parameters (for example, `STimport.ini`).
2. Save the file to the location of your choice.
3. Copy the following list of import parameters and paste into the configuration file:

```
[SilkTestImport]
Execute=
ImportFile=
AllVersions=
ReportResult=
SilentMode=
Projects=
System=
```

```
ProjectAssetResolveConflict=
SystemAssetResolveConflict=
```

4. Set the value of each import parameter. The values do not need to be enclosed in quotes. If the value is not set or the parameter is omitted from the file, the default value is used. Parameter values are not case-sensitive.
5. Save the file.
6. To access the file from the command line, include the `Filename` (f) parameter and the location of the file. For example:

```
C:\>STWImport.exe -u Admin -p admin -d YourDBName -f "C:\STimport.ini"
```

Note: The configuration file header `[SilkTestImport]` must appear at the top of the file, however, you can change the order of the parameters. If you need to comment out a parameter, place a backslash (`\`) character in front of the parameter name. Parameters specified in the configuration file override any identical parameters specified on the command line with one exception. If the `Projects` or `System` parameter is specified both in the configuration file and on the command line, the values for each are used.

Asset Export Command Line Parameters

Use the `STWExport.exe` command line utility to export assets from the command line. Export parameters can be specified entirely from a command line, or unless otherwise noted, specified in a configuration file.

The utility is located in `\Silk\Silk Test\ng\gui\`

Command Line Export Parameters

Syntax	Values	Required	Description
`Username (-u)`	User-defined	Yes	The name of the user used to connect. It can only be specified using the command line. Any configuration file specification for this parameter is ignored. When using Windows authentication, omit the user name.
`Password (-p)`	User-defined	Yes	The password of the user used to connect. This parameter is required if the password is not `Nothing`. It can only be specified using the command line. Any configuration file specification for this parameter is ignored. When using Windows authentication, omit the password.
`DSN (-d)`	User-defined	Yes	The name of the DSN to use to connect to the database. This parameter is required for any DSN names other than "SilkTest". It can only be specified using the command line. Any configuration file specification for this parameter is ignored.
`ExportFile (-ef)`	User-defined	Yes	The file name or location of the ZIP file to be exported.
`Execute (-ex)`	Yes | No (see Note below)	Yes	Commits the export operation. The default value is "No".
`Filename (-f)`	User-defined	No	An INI configuration file that stores command line parameters. It can only be specified using the command line.  **Note:** The INI file parameters override the command line parameters with one exception.

Syntax	Values	Required	Description
			The Projects (-prj) and System (-s) parameter values specified in the INI file do not override the same parameters specified on the command line. Instead the values for each are both used.
`AllVersions (-av)`	Yes | No (see Note below)	No	Exports all versions of the asset(s). The default value is "No".
`ReportResult (-rr)`	User-defined	No	Contains the file name and location of the summary report file to be created. The default location is the current directory.
`SilentMode (-sm)`	Yes | No (see Note below)	No	Prevents the display of the console output.
`Projects (-prj)`	Asset nodes	No	Specifies a single project asset or group of project assets for export. All assets in the database are exported without this parameter. Groups of nodes are separated by the pipe character(|). The format is: `{ProjectName}.{AssetType}.{AssetName}.{Ver#}`. For example: `-prj ProjectA.Visualtest.SampleTest.1` Strings separated by the pipe character (|) must be included within double quotes. For example, `-prj "Common | ProjectB.Result"` **Note:** When specifying the asset type, use the singular form of the asset. For example, use "result" not "results". For asset names comprised of two words (visual test), remove the space (visualtest). There is one exception: test script. Use only the word "script" to specify a test script.
`System (-s)`	Asset nodes	No	Specifies a single project asset or group of project assets for export. All system assets in the database are exported without this parameter. Groups of nodes are separated by the pipe character (|). The format is: `{SystemName}.{SystemType}.{SystemAssetName}`. For example: `-s Options.Record.RecA` Strings separated by the pipe character (|) must be included within double quotes.
ExportAssociated (-ea)	Yes | No (see Note below)	No	To ensure that all referenced assets are exported from every project in the database. The default value is "No".
`OverwriteFile (-of)`	Yes | No (see Note below)	No	Overwrites the existing file (including read-only files) if there is a file name conflict during export. The default value is "Yes".

Note: To specify the default value of a command line parameter, simply omit the parameter from the export command. To specify the non-default value, include only the parameter without its value. For example, to export all versions of an asset, include the `AllVersions` (-av) parameter without a "Yes"

value from your export command. To export only the latest version of an asset, simply omit this parameter to effect the default value of "No".

When specifying parameter values in the export configuration file, the default value is used whenever a parameter or parameter value is omitted. For non-default values, the value must be explicitly stated. For example, to export all versions of an asset, include the `AllVersions` (-av) parameter together with the value "Yes".

Example

```
C:/>STWExport.exe -u Admin -p admin -d MyDBName -mn "c:\MySTAssetFiles" -mt
ds -rr "c:\STExport.txt" -prj ProjectA -ex
```

In the example above, the latest version of asset files in ProjectA are exported to `C:\MySTAssetFiles` and a file is created that contains a summary report of the export.

Example with Configuration File

```
C:\>STWExport.exe -u Admin -p admin -d YourDBName -f "C:\STexport.ini"
```

Creating a Command Line Export Configuration File

To permanently store export settings, you can create a command line export configuration file. Most export parameters can be stored in this file. Several parameters such as `Username`, `Password`, `DSN`, and `Filename` must be specified on the command line.

1. Create an `.INI` file to hold all of the optional export parameters (for example, `STexport.ini`).
2. Save the file to the location of your choice.
3. Copy the following list of export parameters and paste into the configuration file:

   ```
   [SilkTestExport]
   Execute=
   ExportFile=
   AllVersions=
   ReportResult=
   SilentMode=
   Projects=
   System=
   ExportAssociated=
   OverwriteFile=
   ```
4. Set the value of each export parameter. The values do not need to be enclosed in quotes. If the value is not set, the default is used. Parameter values are not case-sensitive.
5. Save the file.
6. To access the file from the command line, include the `FIlename` (-f) parameter and the location of the file. For example:

   ```
   C:\>STWExport.exe -u Admin -p admin -d YourDBName  -f "C:\STexport.ini"
   ```

   Note: The configuration file header `[SilkTestExport]` must appear at the top of the file, however, you can change the order of the parameters. If you need to comment out a parameter, place a backslash (`\`) character in front of the parameter name. Parameters specified in the configuration file override any identical parameters specified on the command line with one exception. If the `Projects` or `System` parameter is specified both in the configuration file and on the command line, the values for each are used.

Logging

During execution from the command line, Silk Test Workbench logs events into .LOG files in the directory `%LOCALAPPDATA%\silk\silktest\logs`. By default, the logging level is set to info and each execution

of a command-line executable creates a new log file that contains the PID, the date, and the time in the filename. Subsequent executions of the same program delete old files and create new ones.

The following table provides a filename example for some command-line executables:

Program	Example Filename
SilkTest.exe	SilkTest(Pid=25184)_20171030_1635.log
STW.exe	STW(Pid=13204)_20171031_0839.log
STWDBConfig.exe	STWDBConfig(Pid=32924)_20171030_1635.log
STWExport.exe	STWExport(Pid=32560)_20171030_1635.log
STWImport.exe	STWImport(Pid=26640)_20171030_1635.log
STWMaint.exe	STWMaint(Pid=13488)_20171030_1618.log
STWPurge.exe	STWPurge(Pid=30096)_20171030_1635.log

You can set your logging preferences by modifying the file `workbench.properties` in the `gui` folder of your Silk Test Workbench installation. By default, this folder should be located under `C:\Program Files (x86)\Silk\SilkTest\ng\gui`. In this file, you can specify the log level, the log style, the log directory, and so on.

For example, if you require multiple log files to be created for a single program, change the setting nativeLog.clearPerProcessLogs to 0 so that the files are not deleted.

Additionally, you can change the setting nativeLog.LogPerProcess to 0 to change the way files are created. In this case all executable files write to a single file named `silktest.log`.

Note: The logger attempts to recreate the file for each execution unless **nativeLog.OverWriteExisting** is set to 0.

Silk Test Workbench Options

You can modify the default options to reflect your testing environment and your preferred way of working. To modify the options, click **Tools** > **Options**.

Global Options

Use **Global** options to set properties for all projects.

Global options include the following:

Asset Management	**Default save behavior**	Set the default save behavior to either create a new version of an asset or replace the current version of the asset each time an asset is saved from a toolbar button, asset dialog box, or shortcut key combination (`Ctrl+S`).
	Maximum asset versions	The maximum number of versions of an asset to keep. Specifying 0 will keep all versions. If the maximum number of versions is reached, and you save a new version of the asset, the oldest version is deleted. When importing an asset, all versions are imported, even if the number of versions exceeds the maximum version number. When saving a new version of such an imported asset, the oldest versions of the asset are deleted to match the specified maximum number of assets.
	Minimum User Access for Purge	Specifies the minimum access level in a project a user needs in order to purge assets in that project. The default value is **Script Writer**. The other available value is **Full Access**. This option can only be changed by Silk Test Workbench users with administrator rights. If there are no projects in the database where the current user has at least the access level specified by the setting, the **File** > **Purge Asset Versions…** item in the Silk Test Workbench menu is disabled. Otherwise, the **Projects** tree in the **Purge Asset Versions** dialog shows the projects in which the user can purge assets.
	Restore Selected Projects in Purge Asset Versions	When set to **Yes**, this option persists the projects that are selected in the **Purge Asset Versions** dialog box. When set to **No**, no projects are selected when the dialog box opens.

eCATT

Tools > **Options** > **Global** > **Integration**

Note: These options are only available when Silk Test Workbench is integrated with eCATT.

RFC Trace	Set to 0n to enable diagnostic tracing of Silk Test Workbench RFC calls. The diagnostic traces are saved in a trace file.

Note: RFC is an SAP API implemented in an SAP .dll called `librfc32.dll` The trace file is created by code in this .dll, which also sets the location to store the trace file. The default location is the system folder.

Set a Windows environment variable for the path that you want RFC trace files to be dumped to. The variable is *RFC_TRACE_DIR* and its value should be the path. This is documented in the RFCSDK help provided by SAP.

General Options

Tools > **Options** > **General**

General options include the following:

Default result view	Select the tab that Silk Test Workbench should display on the **Results** window.
Default wait timeout (milliseconds)	Specify the amount of time that Silk Test Workbench should wait for an object. Default is 5000.
Flash on record	Specify if Silk Test Workbench should flash the Silk Test Workbench window in the taskbar while you record a test.
Minimize on locate	Specify if Silk Test Workbench should minimize the Silk Test Workbench window when you click the **Locate** button.
Minimize on playback	Specify if Silk Test Workbench should minimize the Silk Test Workbench window when you playback a test.
Minimize on record	Specify if Silk Test Workbench should minimize the Silk Test Workbench window when you record a test.
Open result automatically	Specify if Silk Test Workbench should automatically display results after playing back a test.
Save all on playback	Specify if Silk Test Workbench should save all files when you play back a test.
Show Browser Configuration dialog	Specify if Silk Test Workbench should show the **Browser Configuration** dialog so that you can select the browser when you have a browser configuration in your asset. When the value is set to **No**, Silk Test Workbench uses the browser type specified in the application configuration.
Show Local Variables window during playback	Specify if Silk Test Workbench should show the **Local Variables** window during playback and hide it once playback is over.
Show Logic Steps pane in Test Logic Designer wizard	Specify if Silk Test Workbench should show the **Logic Steps** pane in the **Test Logic Designer** wizard. You can also show or hide this pane by clicking the arrow button next to the **Logic Steps** pane title in the wizard.
Show Mobile Configuration Dialog	Specify whether to show the **Mobile Device Configuration** dialog during recording, playback, or while identifying a visual test or .NET script with a mobile device application configuration.
Show Playback Completed dialog	Specify if Silk Test Workbench should show the **Playback Completed** dialog box when test playback finishes.
Show Playback dialog	Specify if Silk Test Workbench should show the **Playback** dialog box before starting a test.
Show Recording Completed dialog	Specify if Silk Test Workbench should show the **Recording Completed** dialog box.

Show Recording Status dialog	Specify if Silk Test Workbench should show the **Record Status** dialog box when you start recording.
Show Test Logic Designer Summary pane	Specify if Silk Test Workbench should show the **Summary** pane in the **Test Logic Designer** wizard. You can also change this setting in the **Summary** pane of the **Test Logic Designer** wizard.
Show wizard welcome screens	Specify if Silk Test Workbench should show a **Welcome** page in each wizard. You can also change this setting in the **Welcome** page of each wizard.

Modifying General Options

The **General** options control high-level behavior when starting up and during record and playback of a visual test or script.

1. Click **Tools** > **Options**.
2. Click **General** in the **Options** menu tree. The general options display in the right side panel.
3. Modify the options by selecting the name of the option in the first column and selecting or entering a new value in the second column.

   When you select an option, the panel below displays a brief explanation.
4. To add a prefix of *Common* to all assets that you add to the Common project, select **Use 'Common' with asset name** from the **Add prefix to Common project assets** list box.
5. Select the tab that you want to display on the **Results** window from the **Default result view** list box.
6. To specify the amount of time that Silk Test Workbench waits for an object, type the time that you want to use in the **Default wait timeout (milliseconds)** text box.
7. To flash the Silk Test Workbench window in the taskbar while you record a test, select **Yes** from the **Flash on record** list box.
8. To minimize the Silk Test Workbench window when you click the **Locate** button, select **Yes** from the **Minimize on locate** list box.
9. To minimize the Silk Test Workbench window when you playback a test, select **Yes** from the **Minimize on playback** list box.
10. To minimize the Silk Test Workbench window when you record a test, select **Yes** from the **Minimize on record** list box.
11. To automatically display results after playing back a test, select **Yes** from the **Open result automatically** list box.
12. To save all files when you playback a test, select **Yes** from the **Save all on playback** list box.
13. To show the **Browser Configuration** dialog so that you can select the browser when you have a browser configuration in your asset, select **Yes** from the **Show Browser Configuration dialog** list box.
14. To show the **Local Variables** window during playback and hide it once playback is over, select **Yes** from the **Show Local Variables window** list box..
15. To show the **Logic Steps** pane in the **Test Logic Designer** wizard, select **Yes** from the **Show Logic Steps pane in Test Logic Designer wizard** list box.

    You can also show or hide this pane by clicking the arrow button next to the **Logic Steps** pane title in the wizard.
16. To show the **Playback Completed** dialog when test playback finishes, select **Yes** from the **Show Playback Completed dialog** list box.
17. To show the **Playback** dialog before starting a test, select **Yes** from the **Show Playback dialog** list box.

    You can assign a result name, description, and options file specifics in the **Playback** dialog box.
18. To show the **Recording Completed** dialog before starting a test, select **Yes** from the **Show Recording Completed dialog** list box.

    You can playback the test, go to the visual test, or save the test in the **Recording Completed** dialog box.

19. To show the **Record Status** dialog box when you start recording, select **Yes** from the **Show Record Status Dialog** list box.
20. To show the **Summary** pane in the **Test Logic Designer** wizard, select **Yes** from the **Show Test Logic Designer Summary pane** list box.

    You can also change this setting in the **Summary** page of the **Test Logic Designer** wizard.
21. To show a Welcome page in each wizard, select **Yes** from the **Show wizard welcome screens** list box.

    You can also change this setting in each wizard's **Welcome** screen.
22. Click **OK**.

Start Screen Options

Click **Tools** > **Options** and then click **Start Screen.**

Use these options to control settings for the **Start Screen**.

Note: You can press `Ctrl+Alt+S`, or click **View** > **Start Screen** at any time to display it.

Show Start Screen when Silk Test Workbench starts	Set to **Yes** to display the **Start Screen** when Silk Test Workbench starts.
Number of recent assets to show.	Select the number of items to display in the **Recently Used Assets** list.

Flags

Assigned to	Use this option to display flags assigned to a selected user. Set to **Current user** to display flags assigned to the current user. Set to **Admin** to display flags assigned to the project administrator. You must have administrator privileges to set this option to **Admin**.  **Note:** If a user is deleted from the system, the user's name is removed from the **Assigned to** list and the flags are reassigned to the project administrator.
Last modified by	Set this option to display flags modified by all users or only display flags that were modified by a specific user. Users who have most recently entered or modified flags against visual tests in the current database display in the list.
Order by date	Set to **Ascending** to display the oldest updated flags at the top of the list. Set to **Descending** to display the most recently updated flags at the top of the list.
Project	Use this option to only display flags created in a selected project. The default setting is to display flags in all projects. Select the Common Project or another project from the list.
Show flags created in	Select an option from the list to display flags created in visual tests, results, or in both visual tests and results.

Modifying Start Screen Options

Use **Start Screen** options to hide the **Start Screen** when Silk Test Workbench starts. You can also determine how flags display in the **Flags** pane of the **Start Screen**. You can also press `Ctrl+Alt+S`, or choose **View** > **Start Screen** at any time to display it.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Start Screen** in the **Options** menu tree. The **Start Screen** options display in the right side panel and the **Flags** subcategory display in the outline view below **Start Screen**. You can set the **Show Start Screen when Silk Test Workbench starts** to **Yes** to display the **Start Screen**

when Silk Test Workbench starts. The **Start Screen** allows users to record and open visual tests for quick testing; perform administrative, test recording and editing tasks; collaborate projects using flags; and get assistance on how to quickly get started creating application tests.

3. In the outline view, click **Flags** to set options.
4. Set the options for your environment.

Assigned to	Use this option to display flags assigned to a selected user. Set to **Current user** to display flags assigned to the current user. Set to **Admin** to display flags assigned to the project administrator. You must have administrator privileges to set this option to **Admin**.
	Note: If a user is deleted from the system, the user's name is removed from the **Assigned to** list and the flags are reassigned to the project administrator.
Last modified by	Set this option to display flags modified by all users or only display flags that were modified by a specific user. Users who have most recently entered or modified flags against visual tests in the current database display in the list.
Order by date	Set to **Ascending** to display the oldest updated flags at the top of the list. Set to **Descending** to display the most recently updated flags at the top of the list.
Project	Use this option to only display flags created in a selected project. The default setting is to display flags in all projects. Select the Common Project or another project from the list.
Show flags created in	Select an option from the list to display flags created in visual tests, results, or in both visual tests and results.

5. Click **OK**.

Record Options

Record options include the following categories:

General	Set asset and mouse move preferences, the object map location, classes to ignore, and whether to resize the application under test (AUT) during recording, to allow the **Silk Recorder** to display next to the AUT.
Locator	Specify the hot key combination to pause recording and the mode you want to use to record locators.
Output	Specify whether controls are captured and properties are recorded, how screens are captured, and whether individual screens are captured during recording.
Hot Keys	Specify what key combinations insert a verification during recording, or start and stop recording.
xBrowser	Specify custom attributes, browser attributes to ignore while recording, and whether to record native user input instead of DOM functions.
WPF	Specify custom attributes and the names of any WPF classes that you want to expose during recording and playback.
Rumba	Specify the objects on a Rumba screen to ignore when creating a screen verification. Click **...** to open the **Edit Excluded Objects** dialog box. Pressing the **+** on the first line adds a new excluded object. The line for each individual object ("Object 1", "Object 2", etc.) has an **x** button that will delete that individual excluded object. How this setting works is that whenever a screen verification is recorded, the excluded objects in settings will be set for that screen verification. The excluded objects in an individual screen

	verification can be edited manually, but this won't update the setting (allowing for the excluded objects to be tailored to an individual verification if necessary).
JavaSWT	Specify the custom attributes that you want to use during locator generation.
SAP	Specify the custom attributes that you want to use during locator generation.
Swing	Specify the custom attributes that you want to use during locator generation.
Win32	Specify the custom attributes that you want to use during locator generation.
WinForms	Specify the custom attributes that you want to use during locator generation.
Open Agent Settings	Custom settings that are directly sent to the agent. Should only be set for very specific testing scenarios, when suggested by support.

Setting Image Asset Recording Preferences

Specify whether you want to record image clicks or screen verifications and set the default value for the image asset accuracy level.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **General**.
4. Set the default accuracy level for new image assets by selecting a value from 1 (low accuracy) to 10 (high accuracy from the **Image asset default accuracy level** list box.
5. To record object clicks, select **Yes** from the **Record image clicks** list box.
6. To record screen verifications, select **Yes** from the **Record screen verifications** list box.
7. Click **OK**.

Setting Object Map Preferences

Specify whether object maps are recorded, which project to record them in, and how to handle locked object maps.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **Locator**.
4. To define whether you want to record object map entries or XPath locators, select the appropriate recording mode from the **Record object map entries** list:
   - **Object map entries for new and existing objects**. This is the default mode.
   - **XPath locators for new and existing objects**.
   - **XPath locators for new objects only**. For objects that already exist in an object map, the object map entry is reused. Choosing this setting enables you to create object maps for the main controls of an AUT, and to persist these object maps while creating additional tests against the AUT.
5. In the **Options** menu tree, click **General**.
6. To record locators for locked object maps, select **Yes** from the **Record locators for locked object maps** list box.

   If you select **Yes**, when an object map is locked, any previously identified object map items are used where applicable and locators are used for any new objects that you record.

   If you select **No**, you must wait for an object map to be unlocked before you can record.
7. To specify the project in which Silk Test Workbench searches for object map items and creates items if no matching items are found, select an option from the **Object map location** list box.

By default, Silk Test Workbench searches the current project to find matching object map items to use when recording. If no matching items are found, Silk Test Workbench searches all referenced projects. If matching items are found in a referenced project, those items are used. If no matching items are found in a referenced project, Silk Test Workbench creates the new items in the current project.

If you select **Current Project**, object map items use the project in which the visual test or script resides.

8. Click **OK**.

Setting Mouse Move Preferences

Specify whether mouse move actions are recorded for Web applications, Win32 applications, and Windows Forms applications that use mouse move events. You cannot record mouse move events for child domains of the xBrowser technology domain, for example Apache Flex and Swing.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **General**.
4. To record mouse move actions, select **Yes** from the **Record mouse move actions** list box.

   Select **Yes** if you are testing a web page that uses mouse move events.

   Silk Test Workbench will only record mouse move events that cause changes to the hovered element or its parent in order to keep scripts short.
5. If you record mouse move actions, in the **Record mouse move delay** text box, specify how many milliseconds the mouse has to be motionless before a `MoveMouse` action is recorded

   By default this value is set to 200.

   Mouse move actions are only recorded if the mouse stands still for this time. A shorter delay will result in more unexpected move mouse actions, a longer delay will require you to keep the mouse still to record an action.
6. Click **OK**.

Setting Classes to Ignore

To simplify the object hierarchy and to shorten the length of the lines of code in your test scripts and functions, you can suppress the controls for certain unnecessary classes in the following technologies:

- Win32.
- Java AWT/Swing.
- Java SWT/Eclipse.
- Windows Presentation Foundation (WPF).

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **General**.
4. In the **Transparent classes** grid, type the name of the class that you want to ignore during recording and playback.

   Separate class names with a comma.
5. Click **OK**.

Setting Locator Preferences

Specify the hot key combination to pause recording and the mode you want to use to record locators.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **Locator**.
4. To set the hot key combination to use to pause recording, select one of the key combinations from the **Identify locator hot keys** list box.

   By default, `Ctrl+Shift` is the hot key combination. However, if you are a Silk Test Classic user, you may prefer to use `Ctrl+Alt`.
5. From the **Identify locator mode** list box, select the mode that you want to use to record locators.

   - **Click** – To identify an object that you want to record by clicking it, use this mode. By default, Silk Test Workbench captures locators using this mode, which provides optimal performance. However, the application does not stop when a locator is captured. As a result, this mode works best when the screen of the application under test does not change rapidly. For example, if you are testing a Web site where multiple objects display simultaneously, use the hot key mode to record the test.
   - **Hot Key** – To identify an object that you want to record by pressing the hot key combination, use this mode. This mode pauses the application when you capture a locator. For example, if you are testing a Web site where multiple windows open and close, use this mode to pause the application to ensure that you capture the correct window. By default the hot key combination is `Ctrl+Shift`.
6. To define whether you want to record object map entries or XPath locators, select the appropriate recording mode from the **Record object map entries** list:

   - **Object map entries for new and existing objects**. This is the default mode.
   - **XPath locators for new and existing objects**. No object map entries are recorded.
   - **XPath locators for new objects only**. For objects that already exist in an object map, the object map entry is reused. Choosing this setting enables you to create object maps for the main controls of an AUT, and to persist these object maps while creating additional tests against the AUT.
7. Click **OK**.

Setting Browser Recording Options

Specify custom attributes, browser attributes to ignore while recording, and whether to record native user input instead of DOM functions.

Silk Test Workbench includes a sophisticated locator generator mechanism that guarantees locators are unique at the time of recording and are easy to maintain. Depending on your application and the frameworks that you use, you might want to modify the default settings to achieve the best results. You can use any property that is available in the respective technology as a custom attribute given that they are either numbers (integers, doubles), strings, item identifiers, or enumeration values.

In xBrowser applications, you can also retrieve arbitrary properties and then use those properties as custom attributes. To achieve optimal results, add a custom automation ID to the elements that you want to interact with in your test.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **xBrowser**.
4. To add a custom attribute for a Web application, in the **Custom attributes** text box, type the attributes that you want to use.

   Using a custom attribute is more reliable than other attributes like caption or index, since a caption will change when you translate the application into another language, and the index might change whenever another object is added before one you have defined already.

Note: To include custom attributes in a Web application, add them to the html tag. For example type, `<input type='button' MyAutomationID='abc' value='click me' />` to add an attribute called `MyAutomationID`.

If more than one object is assigned the same custom attribute value, all the objects with that value will return when you call the custom attribute. For example, if you assign the unique ID, `loginName` to two different text fields, both fields will return when you call the `loginName` attribute.

Note: There is a 62 character limit to attribute names.

5. In the **Locator attribute name exclude list** text box, type the attribute names to ignore while recording.

   Use this list to specify attributes that change frequently, such as size, width, height, and style. You can include the wildcards '*' and '?' in the **Locator attribute name exclude list**.

   For example, if you do not want to record attributes named `height`, add the `height` attribute name to the list.

   Separate attribute names with a comma.
6. In the **Locator attribute value exclude list** text box, type the attribute values to ignore while recording. For example, if you do not want to record attributes assigned the value of `x-auto`, add the `x-auto` attribute value to the list.

   Some AJAX frameworks generate attribute values that change every time the page is reloaded. Use this list to ignore such values. You can also use wildcards in this list.

   Separate attribute names with a comma.
7. To record native user input instead of DOM functions, from the **Record native user input** list box, select **Yes**.

   For example, to record `Click` instead of `DomClick` and `TypeKeys` instead of `SetText`, select **Yes**.

   If your application uses a plug-in or AJAX, specify **Yes** to use native user input. If your application does not use a plug-in or AJAX, we recommend using high-level DOM functions, which do not require the browser to be focused or active during playback. As a result, tests that use DOM functions are faster and more reliable.
8. To enable iframe and frame support for browsers, select **Yes** from the **Enable iframe support** list.

   If you are not interested in the content of the iframes in a web application, disabling the iframe support might improve replay performance. For example, disabling the iframe support might significantly improve replay performance for web pages with many adds and when testing in a mobile browser. This option is ignored by Internet Explorer. This option is enabled by default.
9. Click **OK**.

Setting Custom Attributes

Silk Test Workbench includes a sophisticated locator generator mechanism that guarantees locators are unique at the time of recording and are easy to maintain. Depending on your application and the frameworks that you use, you might want to modify the default settings to achieve the best results. You can use any property that is available in the respective technology as a custom attribute given that they are either numbers (integers, doubles), strings, item identifiers, or enumeration values.

A well-defined locator relies on attributes that change infrequently and therefore requires less maintenance. Using a custom attribute is more reliable than other attributes like caption or index, since a caption will change when you translate the application into another language, and the index might change when another object is added.

For the technology domains listed in the list box on the **Custom Attributes** tab, you can also retrieve arbitrary properties (such as a WPFButton that defines *myCustomProperty*) and then use those properties as custom attributes. To achieve optimal results, add a custom automation ID to the elements that you want to interact with in your test. In Web applications, you can add an attribute to the element that you want to interact with, such as `<div myAutomationId= "my unique element name" />`. Or, in Java SWT, the developer implementing the GUI can define an attribute (for example `testAutomationId`) for a

widget that uniquely identifies the widget in the application. A tester can then add that attribute to the list of custom attributes (in this case, `testAutomationId`), and can identify controls by that unique ID. This approach can eliminate the maintenance associated with locator changes.

If multiple objects share the same attribute value, such as a caption, Silk Test Workbench tries to make the locator unique by combining multiple available attributes with the "and" operation and thus further narrowing down the list of matching objects to a single object. Should that fail, an index is appended. Meaning the locator looks for the *nth* control with the caption *xyz*.

If more than one object is assigned the same custom attribute value, all the objects with that value will return when you call the custom attribute. For example, if you assign the unique ID, `loginName` to two different text fields, both fields will return when you call the `loginName` attribute.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click the technology domain for the application that you are testing.
   For example, if you are testing an SAP application, click **SAP** in the menu tree.

   Note: You cannot set custom attributes for Flex or Windows API-based client/server (Win32) applications.
4. In the **Custom attributes** text box, type the attributes that you want to use.

   If custom attributes are available, the locator generator uses these attributes before any other attribute. The order of the list also represents the priority in which the attributes are used by the locator generator. If the attributes that you specify are not available for the objects that you select, Silk Test Workbench uses the default attributes for the application that you are testing.

   Separate attribute names with a comma.

   Note: To include custom attributes in a web application, add them to the html tag. For example type, `<input type='button' bcauid='abc' value='click me' />` to add an attribute called `bcauid`.

   Note: To include custom attributes in a Java SWT control, use the `org.swt.widgets.Widget.setData(String key, Object value)` method.

   Note: To include custom attributes in a Swing control, use the `putClientProperty("propertyName", "propertyValue")` method.
5. Click **OK**.

Setting WPF Classes to Expose During Recording and Playback

Silk Test Workbench filters out certain controls that are typically not relevant for functional testing. For example, controls that are used for layout purposes are not included. However, if a custom control derives from an excluded class, specify the name of the related WPF class to expose the filtered controls during recording and playback.

Specify the names of any WPF classes that you want to expose during recording and playback. For example, if a custom class called *MyGrid* derives from the WPF `Grid` class, the objects of the *MyGrid* custom class are not available for recording and playback. `Grid` objects are not available for recording and playback because the `Grid` class is not relevant for functional testing since it exists only for layout purposes. As a result, `Grid` objects are not exposed by default. In order to use custom classes that are based on classes that are not relevant to functional testing, add the custom class, in this case *MyGrid*, to the **OPT_WPF_CUSTOM_CLASSES** option. Then you can record, playback, find, verify properties, and perform any other supported actions for the specified classes.

1. Click **Tools** > **Options**.

2. Click the plus sign (+) next to **Record** in the **Options** menu tree.
3. Click **WPF**.
4. In the **Custom WPF class names** grid, type the name of the class that you want to expose during recording and playback.

   Separate class names with a comma.
5. Click **OK**.

Setting Record Output Options

Modify **Record Output** options to specify whether controls are captured and properties are recorded, how screens are captured, and whether individual screens are captured during recording.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click the plus sign (+) next to **Output**. The **Visual test** subcategory displays.
4. Click **Visual test**.
5. Modify options in the options panel by selecting the name of the option in the first column and selecting or entering a new value in the second column.

   When you select an option, the panel below displays a brief explanation of it.

   You can set the following output options:

Screen capture	Determines how Silk Test Workbench captures test application screens during recording. Set to **Application** to capture the test application and any windows within the test application. Set to **Active Window** to capture only the test application's active window during recording. Set to **Desktop** to capture screens of all applications, including the test application, that are visible in the desktop during recording. Set to **None** to capture no screens during recording.
Screen capture (test steps)	Set to **Yes** to capture individual screens that display in **Screen Preview**. This enables you to view the actions that occur on individual screens in **Screen Preview** in conjunction with individual test steps in the **Test Steps** pane. Additionally, you can compare the screens that are recorded during playback against the screens captured when the visual test was first recorded if test step screens are captured during playback also. Set to **No** to capture no individual test steps during recording. If the **Screen capture** setting is set to anything other than **None**, a screen is captured for the main context of the group of actions that occur on a dialog box or Web page. As a result, each individual test step is associated with a group screen in **Screen Preview**.
Control capture	Set to **Yes** to capture controls and their properties during recording. The controls can then be selected using **Screen Preview** without requiring access to the test application. To improve playback performance, set this option to **No**. When recording Web applications the following attributes are stored with the captured controls: • `id` • `class` • `name` • `value`

- `href`
- `innerText`

To add additional attributes, use the **Custom attributes** field under **Tools** > **Options** > **Record** > **xBrowser**.

Note: Attribute text values are truncated to the length specified in the **Maximum attribute value length** field under **Tools** > **Options** > **Record** > **xBrowser**.

Browser control capture

Defines whether controls and their properties are captured when recording against web browsers. The default value is **No**. Set to **Yes** to work with controls and their properties in **Screen Preview** without requiring access to the web browser. Silk Test Workbench ignores this option if the **Control capture** option is set to **No**.

Warning: If you set this option to **Yes**, playing back your tests might take longer. For improved playback performance, leave this option on **No**.

Tip: Use these options in conjunction with the **Playback Results: Visual test** options to compare playback results to the screens that were captured when the visual test was first recorded.

6. *Optional*: Click **Save As** to save your settings for the **Playback** options as a group.
7. Click **OK**.

Record Output: Visual Test Options

Use the **Record Output: Visual Test** options to control behavior while recording controls and screens in visual tests. For more information about using these options, see *Limiting Database Growth*.

You can set the following output options:

Screen capture

Determines how Silk Test Workbench captures test application screens during recording.

Set to **Application** to capture the test application and any windows within the test application.

Set to **Active Window** to capture only the test application's active window during recording.

Set to **Desktop** to capture screens of all applications, including the test application, that are visible in the desktop during recording.

Set to **None** to capture no screens during recording.

Screen capture (test steps)

Set to **Yes** to capture individual screens that display in **Screen Preview**. This enables you to view the actions that occur on individual screens in **Screen Preview** in conjunction with individual test steps in the **Test Steps** pane. Additionally, you can compare the screens that are recorded during playback against the screens captured when the visual test was first recorded if test step screens are captured during playback also.

Set to **No** to capture no individual test steps during recording. If the **Screen capture** setting is set to anything other than **None**, a screen is captured for the main context of the group of actions that occur on a dialog box or Web page. As a result, each individual test step is associated with a group screen in **Screen Preview**.

Control capture

Set to **Yes** to capture controls and their properties during recording. The controls can then be selected using **Screen Preview** without requiring access to the test application. To improve playback performance, set this option to **No**.

When recording Web applications the following attributes are stored with the captured controls:

- `id`
- `class`

- `name`
- `value`
- `href`
- `innerText`

To add additional attributes, use the **Custom attributes** field under **Tools** > **Options** > **Record** > **xBrowser**.

Note: Attribute text values are truncated to the length specified in the **Maximum attribute value length** field under **Tools** > **Options** > **Record** > **xBrowser**.

Browser control capture

Defines whether controls and their properties are captured when recording against web browsers. The default value is **No**. Set to **Yes** to work with controls and their properties in **Screen Preview** without requiring access to the web browser. Silk Test Workbench ignores this option if the **Control capture** option is set to **No**.

Warning: If you set this option to **Yes**, playing back your tests might take longer. For improved playback performance, leave this option on **No**.

Setting Record Hot Key Options

Modify the **Hot Keys** options to specify what key combinations start and stop recording or insert a verification during recording.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **Hot Keys**.

   You can set hot key combinations for the following:

Insert Verification	Specifies the key combination to temporarily suspend recording and displays the **Test Logic Designer** wizard. Select the object that you want to verify after the wizard opens.
Insert Verification using the current object	Specifies the key combination to temporarily suspend recording and selects the currently highlighted object for verification. With this key combination, the **Test Logic Designer** wizard opens with the selected object set as the control to verify.
Start / Stop Record	Specifies the key combination used to toggle recording on and off.

4. To change the hot key combination, perform the following steps:
   a) In the option panel to the right, select the hot key option you want to change.
   b) Click the ellipsis button next to the selected operation. The **Define Hotkey** dialog box opens.
   c) To use a modifier key, check the appropriate check box.
   d) From the **Key** list, select the key that you want to use as the hot key.

      Key values include A-Z, 0-9, F1-F12.

      Silk Test Workbench enters the key sequence in the text box.
   e) Click **OK**.
5. Click **OK**.

Creating a Record Options Profile

To quickly configure the recording environment, you can change the default values of any record option and save the changes to each modified option in a record options profile. After creating a profile, you can apply this set of record options before running a test script or visual test.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **General**, **Locator**, **xBrowser**, **SWT**, **Output**, or **Hot Keys** and then modify the **Record** options to change the way Silk Test Workbench identifies and captures your application's behavior during recording.
4. Click **Save As**. The **Record Settings** dialog box opens.
5. In the **Name** text box, enter a name for the current set of record option settings, or select an existing group from the list.

   If you select an existing group, the current settings replace the ones previously saved under this name.
6. In the **Owner** text box, select the user for these settings.
7. Click **OK**.

Applying Saved Record Options

You must create a record options profile prior to applying it.

Save a set of recording options in a profile and later apply them as a group. You can use this feature to maintain different sets of recording options for particular environments.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Record** in the **Options** menu tree. The **Record** options display in the right side panel.
3. Click **General**, **Locator**, **xBrowser**, **SWT**, **Output**, or **Hot Keys**.
4. From the **Record Settings** list, select the name of the saved options you want to apply or **System Defaults** to restore the original settings.

   Note: If you modify an existing options profile, ensure that you update the options profile correctly. To change an existing options profile, you must click **Save As** with the options set to the values that you want to use, and then specify the appropriate options profile name. Otherwise, the selected options are saved in the dialog box, but not applied to the options profile.

   Silk Test Workbench updates the current **Record** options with the settings in the saved profile.
5. Click **OK**.

Playback Options

Playback options include the following:

General	Specify when an object is called, whether an object is highlighted during replay, whether unresponsive applications are closed, the playback mode, whether to show script automation calls in the **Output** window, and the amount of time to wait for applications to become active.
Timing	Specify the number of milliseconds to wait for an application to become ready, the number of milliseconds to wait before taking a screen shot of the window, the time that may elapse between capturing two enumerations, the delay between keyboard strokes during replay, the delay before each mouse event in a script, the time to wait for an object to be resolved during replay, and the time for canceling pending playback actions.
Results	Set the default result name, result numbering formula, pass criteria percentage, whether to save all information, and whether to capture controls and screens in visual tests during playback.

Hot Keys	Set the default save behavior to either create a new version of an asset or replace the current version of the asset each time an asset is saved from a toolbar button, asset dialog box, or shortcut key combination (`Ctrl+S`).
xBrowser	Specify any URLs to exclude from synchronization, the synchronization mode, and timeout values for Web applications.
Close	Specify the list of buttons, menu items, confirmation dialog boxes, and key strokes that close windows and dialog boxes.
Playback Status Dialog	Specify whether to display the **Playback Status** dialog box during test playback, and whether to display screenshots in the dialog box.
Open Agent Settings	Custom settings that are directly sent to the agent. Should only be set for very specific testing scenarios, when suggested by support.

Setting General Playback Options

Specify when an object is called, whether an object is highlighted during replay, whether hanging applications are closed, the playback mode, whether to show script automation calls in the **Output** window, and the amount of time to wait for applications to become active.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
3. Click **General**.

Catch Exception when Thrown	The purpose of this setting is for you to be able to step into an exception when it occurs. If you are creating and testing your script, set it to **Yes** so that you can see where the issue is. If you are running automated scripts, you should have your own error handling and mark this setting as **No**. If you have error handling turned on in your script, you will have to click **Continue** to get to it. If the setting is **No** and you do not have your own error handling, you will only be able to **End** the script. When set to **Automatic**, if Silk Test runs normally, then the setting acts as if it is set to **Yes**. If Silk Test is running scripts executed via Silk Central Test Manager, it acts as if it is set to **No**. This way if a script has its own error handling and is run via automation, the script will continue to work.
Close unresponsive applications	Shuts down the application if communication between the Silk Test Agent and the application fails or times out. When testing applications that cannot run multiple instances, specify **Yes**.
Ensure object is active	Ensures that the object is active before a call is executed.
Execute screen verifications	Controls if screen verifications are recorded and played back for Rumba.
Highlight object during playback	Highlights the current object during replay select.
Playback mode	Select one of the following options: **Default** — Use this mode for the most reliable results. By default, each control uses either the mouse and keyboard (low level) or API (high level) modes. With the default mode, each control uses the best method for the control type. **High level** — Use this mode to replay each control using the API.

Low level	Use this mode to replay each control using the mouse and keyboard.
Show automation actions in the Output window	Show automation calls for scripts in the **Output** window. You must open the **Output** window to see the results. Click **View** > **Output**.
Show properties in Local Variables window	This setting controls whether the properties of a class are shown in the **Local Variables** window along with normal member variables and parameters. Since properties of a class may contain implementation code (calculations), sometimes these can contain errors which cause problems displaying their values. In addition, the calculation of a property may take longer than the retrieval of a member variable. The default value is **Yes**.

4. Click **OK**.

Setting xBrowser Synchronization Options

Specify the synchronization and timeout values for Web applications. Synchronization is performed automatically before and after every method call. A method call is not started and does not end until the synchronization criteria is met.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
3. Click **xBrowser**.
4. In the **Synchronization exclude list** text box, type the entire URL or a fragment of the URL for any service or web page that you want to exclude.

   Some AJAX frameworks or browser applications use special HTTP requests, which are permanently open in order to retrieve asynchronous data from the server. These requests may let the synchronization hang until the specified synchronization timeout expires. To prevent this situation, either use the HTML synchronization mode or specify the URL of the problematic request in the **Synchronization exclude list** setting.

   For example, if your web application uses a widget that displays the server time by polling data from the client, permanent traffic is sent to the server for this widget. To exclude this service from the synchronization, determine what the service URL is and enter it in the exclusion list.

   For example, you might type:

   - http://example.com/syncsample/timeService
   - timeService
   - UICallBackServiceHandler

   Separate multiple entries with a comma.

   Note: If your application uses only one service, and you want to disable that service for testing, you must use the HTML synchronization mode rather than adding the service URL to the exclusion list.

   Tip: Micro Focus recommends adding a substring of an URL to the exclude list, instead of the entire URL. For example, add `/syncsample` to the exclude list instead of `http://example.com/syncsample/timeService`. Excluding the entire URL might not work because the browser might return only a relative URL to Silk Test Workbench. For example, if the browser returns only `/syncsample/timeService` and you have added `http://example.com/syncsample/timeService` to the exclude list, Silk Test Workbench does not exclude the returned URL.
5. From the **Synchronization mode** list box, select the synchronization algorithm for the ready state of a Web application.

The synchronization algorithm configures the waiting period for the ready state of an invoke call.

Using the **HTML** mode ensures that all HTML documents are in an interactive state. With this mode, you can test simple Web pages. If more complex scenarios with Java script are used, it might be necessary to manually script synchronization functions.

Using the AJAX mode eliminates the need to manually script synchronization functions (such as wait for objects to appear or disappear, wait for a specific property value, and so on), which eases the script creation process dramatically. This automatic synchronization is also the base for a successful record and playback approach without manual script adoptions.

6. In the **Synchronization timeout** text box, enter the maximum time, in milliseconds, to wait for an object to be ready. By default this value is set to 300000.
7. In the **Whitelist for iframe support**, specify attributes of iframes and frames that should be considered during testing.

   Every entry in the list defines an attribute name and the corresponding value. All iframes and frames that do not match at least one of the entries are excluded. Wildcards are allowed, for example the entry "name:*form" would include <IFRAME name="user-form" src=…>. This option is ignored by Internet Explorer. If the list is empty, all iframes and frames are considered during testing. Separate multiple entries with a comma.
8. In the **Blacklist for iframe support**, specify attributes of iframes and frames that should be excluded during testing.

   Every entry in the list defines an attribute name and the corresponding value. All iframes and frames that do not match at least one of the entries are considered during testing. Wildcards are allowed, for example the entry "src:*advertising*" would exclude <IFRAME src=http://my.domain/advertising-banner.html>. This option is ignored by Internet Explorer. If the list is empty, all iframes and frames are considered during testing. Separate multiple entries with a comma.
9. Click **OK**.

Setting Playback Timing Options

You can use the **Playback** options to control timing during playback.

1. Click **Tools** > **Options**. The **Options** dialog box opens.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** options display in the right side panel and the **Timing** subcategory displays in the outline view below **Playback**.
3. Click **Timing**.
4. In the **Application ready timeout** text box, specify the number of milliseconds to wait for an application to become ready.

   If the application is not ready within the specified timeout, Silk Test Workbench raises an exception.
5. In the **Keyboard event delay** text box, specify the delay, in milliseconds, between keyboard strokes during replay. The optimal number you select can vary, depending on the application that you are testing. For example, if you are testing a web application, a setting of 1 millisecond radically slows down the browser. However, setting this to 0 (zero) may cause basic application testing to fail.
6. In the **Mouse event delay** text box, specify the delay, in milliseconds, used before each mouse event in a script.

   The delay affects moving the mouse, pressing buttons, and releasing buttons.
7. In the **Object enabled timeout** text box, specify the time, in milliseconds, to wait for an object to become enabled during replay.
8. In the **Object resolve timeout** text box, specify the time, in milliseconds, to wait for an object to be resolved during replay.
9. Specify a value for the **Screen shot delay**.

   Enter the number of milliseconds to wait once focus has been set to a new window before taking a screen shot of the window.

   By default this value is set to 300 milliseconds.

10. In the **Unresponsive application timeout** text box, enter the time, in milliseconds, for canceling pending playback actions.
11. *Optional*: Click **Save As** to save your settings for the **Playback** options as a group.
12. Click **OK**.

Setting Close Options

Specify the list of buttons, menu items, confirmation dialog boxes, key strokes that close windows and dialog boxes, and the amount of time before a window close strategy is executed.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
3. Click **Close**.
4. In the **Buttons to close windows** text box, specify the list of buttons used to close windows closed with `Close`, `CloseWindows`, and `Exit`.
5. In the **Buttons to confirm dialogs** text box, specify the list of buttons used to close confirmation dialog boxes (dialog boxes that appear when closing windows with `Close`, `CloseWindows`, and `Exit`).
6. In the **Keys to close dialogs** text box, specify the keystroke sequence used to close dialog boxes (used by `Close`, `CloseWindows`, and `Exit`).
7. In the **Menu items to close windows** text box, specify the list of menu items used to close windows with `Close`, `CloseWindows`, and `Exit`.
8. Specify a value for the **Close window timeout**.

   Enter the time in milliseconds to wait before trying the next close strategy. Before failing, four close attempts are executed, so the total time before a close fails is four times the value you specify.
9. Click **OK**.

Setting Playback Result Options

Modify **Playback Results** options to set the default result name, result numbering formula, pass criteria percentage, and whether to save all information.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
3. Click **Results**. The **Results** options appear in the right pane.
4. Set the options for your environment:

Default result name	Enter a name to use as the result name for every result. Leave this field empty to use the name of the visual test or test script as the result name.
Result option	Select from the following list: • **Save results with a new run number** – Results from the playback of a visual test or test script are saved with a new incremental run number. For example, a new result is saved with a run number of 1. Subsequent results for the same visual test or test script are saved with run numbers of 2, 3, 4, etc. This is the default selection. • **Append results to current run number** – Results from the current playback are appended to the results of the latest existing run number. A new run number is not created. This option only appears with an existing result. • **Overwrite existing run number with new results** – Results from the current playback overwrite the results of the latest existing run number. A new run number is not created. This option only appears with an existing result.

- **Do not save this result** – Results are not created when playing back a visual test or test script.

Result pass criteria (percentage)
Sets a user-defined percentage of passed verifications as the criteria to define the success of all future runs. For example, a result pass criteria of 90% means that at least 9 out of 10 verifications in a visual test must pass for the result of the playback to pass.

Save all information
Determines whether to save all information in a result about the playback of a visual test or only summary and error information.

5. *Optional*: Click **Save As** to save your settings for the **Playback** options as a group.
6. Click **OK**.

Setting Playback Result Options for Visual Tests

Use the **Playback Results: Visual Test** options to control capturing controls and screens in visual tests during playback. For more information about using these options, see *Limiting Database Growth*.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
3. Click the plus sign (+) next to **Results** and then click **Visual test**. The **Visual test** options appear in the right pane.
4. Set the options for your environment:

Control capture
Capture controls and their properties during playback.

Set to **Same as recorded** to use the same setting as the **Control capture** setting that existed in the **Record Output** options when the visual test was recorded.

Set to **Yes** to capture controls and their properties during playback. The controls can then be selected using **Screen Preview**.

Note: If the **Screen capture** setting is set to **None** for recording, you must set **Control capture** to **Yes** and the playback **Screen capture** setting to a value other than **None** to capture controls for playback.

Set to **No** to improve playback performance.

Screen capture
Determines how Silk Test Workbench captures test application screens during playback.

Set to **Same as recorded** to use the same setting as the **Screen capture** setting that existed for each specific screen when the visual test was recorded. For example, visual tests containing screens recorded with both **Application** and **Active Window** settings will playback using these settings as they apply to each specific screen in the visual test.

Set to **Application** to capture the test application and any windows within the test application.

Set to **Active Window** to capture only the test application's active window during playback.

Set to **Desktop** to capture screens of all applications, including the test application, that are visible in the desktop during playback.

Set to **None** to disable the capture of screens during playback.

Note: When this setting is set to **None**, no controls or screens are captured in the **Results** window, regardless of the values specified in the **Control capture** and **Screen capture (test steps)** settings.

Screen capture (test steps)

Set to **Yes** to capture individual screens that display in **Screen Preview**. This enables you to view the actions that occur on individual screens in **Screen Preview** in conjunction with individual test steps in the **Test Steps** pane. Additionally, you can compare the screens that are recorded during playback against the screens captured when the visual test was first recorded if test step screens were captured during recording also.

Set to **No** to capture no individual test steps during playback. If the **Screen capture** setting is set to anything other than **None**, the group screen associated with the individual test step is displayed in **Screen Preview**.

Tip: Use these options in conjunction with the **Record Output** options to compare playback results to the screens that were captured when the visual test was first recorded.

5. *Optional*: Click **Save As** to save your settings for the **Playback** options as a group.
6. Click **OK**.

Setting Playback Result Options for .NET Scripts

Use the **Playback Results: .NET Script** options to control the capturing of screens during the playback of .NET scripts.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
3. Click the plus sign (+) next to **Results** and then click **.NET Script**. The **.NET Script** options display in the right pane.
4. From the **Screen capture** list, select how Silk Test Workbench should capture test application screens during playback:
   - Set to **None** to disable the capture of screens during playback.
   - Set to **Application** to capture the test application and any windows within the test application.
   - Set to **Active Window** to capture only the test application's active window during playback.
   - Set to **Desktop** to capture screens of all applications, including the test application, that are visible in the desktop during playback.
5. *Optional*: Click **Save As** to save your settings for the **Playback** options as a group.
6. Click **OK**.

Setting Playback Result Options for Keyword-Driven Tests

Use the **Playback Results: Keyword-Driven Test** options to control the capturing of screens during the playback of keyword-driven tests.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
3. Click the plus sign (+) next to **Results** and then click **Keyword-Driven Test**. The **Keyword-Driven Test** options display in the right pane.
4. From the **Screen capture** list, select how Silk Test Workbench should capture test application screens during playback:
   - Set to **None** to disable the capture of screens during playback.
   - Set to **Application** to capture the test application and any windows within the test application.
   - Set to **Active Window** to capture only the test application's active window during playback.
   - Set to **Desktop** to capture screens of all applications, including the test application, that are visible in the desktop during playback.
5. *Optional*: Click **Save As** to save your settings for the **Playback** options as a group.
6. Click **OK**.

Modifying Playback Hot Keys

Modify the Playback Hot Key to stop the playback while Silk Test Workbench is minimized.

In the Options menu tree, click the plus sign (+) next to **Playback** and then click **Hot Keys**. Select **Stop playback** and edit the key specification. By default, the hot key to stop playback is `Alt+F12`.

Creating a Playback Options Profile

You can save a set of playback options in a profile and later apply them as a group. You can use this feature to maintain different sets of playback options for particular environments.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
3. Click **Timing**, **Results**, **Visual test**, or **Hot Keys** and then modify the **Playback** options to change the way Silk Test Workbench identifies and captures your application's behavior during playback.
4. Click **Save As**. The **Playback Settings** dialog box opens.
5. In the **Name** text box, enter a name for the current set of playback option settings, or select an existing group from the drop-down list. If selecting an existing group, the current settings replace the ones previously saved under this name.
6. In the **Owner** text box, select the user for these settings.
7. Click **OK**.

Applying Saved Playback Options

You must create a playback recording options profile prior to applying it.

Restore the default playback options or a new profile that you have created to playback tests to meet your standards.

1. Click **Tools** > **Options**.
2. Click the plus sign (+) next to **Playback** in the **Options** menu tree. The **Playback** subcategories appear.
3. Click **Timing**, **Results**, **Visual test**, or **Hot Keys**.
4. From the **Playback Settings** list at the bottom of the panel, select the name of the saved options you want to apply or **System Defaults** to restore Silk Test Workbench's original settings.

   Note: If you modify an existing options profile, ensure that you update the options profile correctly. To change an existing options profile, you must click **Save As** with the options set to the values that you want to use, and then specify the appropriate options profile name. Otherwise, the selected options are saved in the dialog box, but not applied to the options profile.

   Silk Test Workbench updates the current **Playback** options with the settings in the saved profile.
5. Click **OK** to make the new option settings active.

Scripting Options

The **Scripting** options control the layout of scripts including font type and size.

1. Click **Tools** > **Options**.
2. Click **Scripting** in the **Options** tree. The scripting options display in the right side panel.
3. In the **Font** text box, click the ellipsis button. The **Font** dialog box opens.
4. Select the font type, style, and size that you want to use for scripts and then click **OK**.

5. Click **OK**.

Setting Advanced Options

Set advanced options to enable fallback support for hybrid apps, to specify whether locator attribute names should be case sensitive, and so on. Modify the **Advanced** > **ActiveData** options to control aspects of how Silk Test Workbench writes data to text-based (.csv or .txt) ActiveData files when editing the files using the **ActiveData Asset Setup** window.

1. Click **Tools** > **Options**. The **Options** dialog box opens.
2. Click the **Advanced** tab. The **Advanced Options** page displays.
3. To test an embedded Chrome application, specify the executable and the port as a value pair in the **Enable embedded Chrome support** field.
   For example, `myApp.exe=9222`.

   To specify multiple embedded Chrome applications, separate the value pairs with a comma.
4. Enable **Fallback support for web views on Android and iOS** to enable the mobile native fallback support for hybrid mobile applications that are not testable with the default browser support.
5. Enable **Microsoft Accessibility** to enable Microsoft Accessibility in addition to the normal Win32 control recognition.
6. Enable **Remove focus on capture text** to remove the focus from the window before capturing a text.

   A text capture is performed during recording and replay by the following methods:

   - `TextClick`
   - `TextCapture`
   - `TextExists`
   - `TextRect`
7. Enable **Locator attribute are case sensitive** to set locator attribute names to be case sensitive. The names of locator attributes for mobile web applications are always case insensitive, and this option is ignored when recording or replaying mobile web applications.
8. To force Mozilla Firefox to open external links in a new tab instead of a new window, enable **Use single-window mode for Firefox**.

   Note: This option only works with Mozilla Firefox 52 or later.
9. Click the plus sign (+) next to **Advanced** in the **Options** menu tree and then click **ActiveData**. The **ActiveData** options display in the right side panel.
10. Set the options for your environment:

Always quote fields

Select **Yes** to enclose the data in each cell of a new ActiveData file in quotes. This option pertains to new ActiveData files only. Setting this option to **Yes** has no affect on existing files.

Set this option to **Yes** when data to be used from an ActiveData file requires quotes.

Note: If set to **Yes**, data in new ActiveData files does not appear in quotes during editing. However, the quotes do appear when editing the ActiveData file using the file's native application.

Write empty trailing fields

Set to **Yes** if you want trailing spaces entered after characters for each data item. For example, if the required data in a field is "data ", setting this option to **Yes** includes the trailing spaces as part of the data, so "data" is written to the field in the ActiveData file. Selecting **No** only includes characters without any trailing spaces. For example, if set to **No**, "data " written into a field would be saved in the ActiveData file as "data".

Set this option to **Yes** if data to be used in an ActiveData test requires the trailing spaces to correctly test the application.

11. Click **OK**.

Advanced ActiveData Options

Modify the **Advanced ActiveData** options to control aspects of how Silk Test Workbench writes data to text-based (.csv or .txt) ActiveData files when editing the files using the **ActiveData Asset Setup** window.

You can set the following options for ActiveData:

Always quote fields

Select **Yes** to enclose the data in each cell of a new ActiveData file in quotes. This option pertains to new ActiveData files only. Setting this option to **Yes** has no affect on existing files.

Set this option to **Yes** when data to be used from an ActiveData file requires quotes.

Note: If set to **Yes**, data in new ActiveData files does not appear in quotes during editing. However, the quotes do appear when editing the ActiveData file using the file's native application.

Write empty trailing fields

Set to **Yes** if you want trailing spaces entered after characters for each data item. For example, if the required data in a field is "data ", setting this option to **Yes** includes the trailing spaces as part of the data, so "data" is written to the field in the ActiveData file. Selecting **No** only includes characters without any trailing spaces. For example, if set to **No**, "data " written into a field would be saved in the ActiveData file as "data".

Set this option to **Yes** if data to be used in an ActiveData test requires the trailing spaces to correctly test the application.

Silk Test Workbench Administration

Provides procedures for administering Silk Test Workbench databases, projects, and users.

Do I Need Administrator Privileges to Run Silk Test Workbench?

You require the following privileges to install or run Silk Test Workbench:

- To install Silk Test Workbench, you must have local administrator privileges.
- To install Silk Test Workbench on a Windows server, you must have domain-level administrator privileges.
- To run Silk Test Workbench, you require full access rights to the following folders, including all subfolders:
  - `C:\ProgramData\Silk\SilkTest`.
  - `%APPDATA%\Roaming\Silk\SilkTest`.
  - `%APPDATA%\Local\Silk\SilkTest`.
  - `%TEMP%`.

Configuring a Silk Test Workbench Database

To store any assets and settings, Silk Test Workbench requires a database. The default Silk Test Workbench installation, includes a Microsoft Access database designed for single user operation. For professional enterprise operation, Silk Test Workbench supports Microsoft SQL Server and Oracle databases. To use Microsoft SQL Server or Oracle databases with Silk Test Workbench, create and configure the database on a central server machine and, on each client machine, create a data source name (DSN) that targets the central database. Micro Focus recommends using Microsoft SQL Server for long term automation projects with multiple users contributing to the same project.

For information about the supported databases, refer to the *Release Notes*.

Note: If a database is updated when a new version of Silk Test Workbench is installed, all Silk Test Workbench users that use this database need to update to the new version of Silk Test Workbench.

Note: Silk Test cannot use 64-bit DSNs. You can review the default DSNs and ODBC drivers for 64-bit machines using the WOW64 tools, which are located at `C:\Windows\SysWow64`.

Configuring an SQL Server Database

Important: We recommend that a database administrator or person with general knowledge of database administration performs the database setup and configuration.

Silk Test Workbench stores and accesses test assets stored in a database configured for use with Silk Test Workbench. This section describes how to configure the following SQL Server databases to use as a Silk Test Workbench database.

For information about new features, supported platforms, and tested versions, refer to the *Release Notes*.

SQL Server Requirements

A SQL Server database configured for use with Silk Test Workbench has the following requirements:

- Each computer running Silk Test Workbench must be able to access the computer where the SQL Server database resides and have a new data source configured.
- Silk Test Workbench must be installed on all computers accessing the database.
- The Silk Test Workbench database connection file must be configured to point to the SQL Server database.
- The user that is used to connect to the SQL Server database must have the `VIEW_SERVER_STATE` permission on the database.

Note: Silk Test cannot use 64-bit DSNs. You can review the default DSNs and ODBC drivers for 64-bit machines using the WOW64 tools, which are located at `C:\Windows\SysWow64`.

For additional information about maintaining a Silk Test Workbench database, refer to the *Silk Test Workbench Help*.

Creating a New SQL Server Database

This section describes how to create a SQL Server database using SQL Server Management Studio. In addition to the following procedures, you must also populate the database with Silk Test Workbench tables using the Silk Test Workbench Database Maintenance utility.

1. In the Object Explorer of SQL Server Management Studio, right-click the **Databases** folder and choose **New Database**. The **New Database** dialog box opens.
2. Enter a name for the database in the **Database name** text box.
3. Click **OK**.

Creating a New SQL Server Admin User

A SQL Server user with system administrator rights is required to perform subsequent setup steps.

This user requires the following permissions:

- Either `sysadmin` or both `db_ddladmin` and `db_owner`.
- `db_datareader`.
- `db_datawriter`.

These permissions are required because the user needs to be able to create indexes by using the *CREATE INDEX (Transact-SQL)* statement. Once the Silk Test Workbench database has been created, all users who connect to the database only require `db_datareader` and `db_datawriter` permissions.

1. To create a new user with system administrator rights, perform the following steps:
   a) In the Object Explorer of SQL Server Management Studio, navigate to the **Security** folder and expand it.
   b) Right-click the **Logins** folder and choose **New Login**. The **Login - New** dialog box opens.
   c) Select the **General** page, and then enter a user name in the **Login name** text box.
   d) Select SQL Server Authentication and enter a password.
   e) Select the default database from the **Default database** list.
   f) Select the **Server Roles** page, and then check the **sysadmin** check box in the **Server roles** list.
   g) Select the **User Mapping** page.
   h) In the **Map** column, check the check box for the database that the new login can access.
   By default, the login name appears in the **User** column. Leave this value.
   i) In the **Database role membership for** list, check the **db_owner** check box.
   j) Click **OK**.
2. To create a new schema for the new user, perform the following steps:
   a) In the **Object Explorer**, navigate to the database in which you want to create a schema.
   b) Expand the database object tree to show the **Schemas** folder.
   The **Schemas** folder is a child of the **Security** folder for the database.

For example, if you are adding a schema to the *master* database, expand the following folders to show the **Schemas** folder: **Databases** > **System Databases** > **master** > **Security**.

c) Right-click the **Schemas** folder and choose **New Schema**.

d) In the **Schema name** text box, type a name for the new schema.

The new schema name must match the name of the previously created user with system admin rights.

e) Assign the new user with system administrator rights as the owner of the schema.

f) Click **OK**.

3. To add the schema to the database, perform the following actions:

a) Expand the database object tree to show the **Users** folder.

The **Users** folder is a child of the **Security** folder for the database.

For example, if you are adding a schema to the *master* database, click **Databases** > **System Databases** > **master** > **Security** > **Users**.

b) Double-click the admin user you have just created. The **Database User** dialog box appears.

c) Select the **Owned Schemas** page and check that the schema you have just created has been assigned to the new database.

Setting Up Users in SQL Server

The following procedure should be performed on the database server computer for each user who needs to connect to the SQL server database.

1. In the Object Explorer of SQL Server Management Studio, navigate to the **Security** folder and expand it. Right-click the **Logins** folder and choose **New Login**. The **Login - New** dialog box opens.
2. Select the **General** page, and then enter a name for the database in the **Login name** text box.
3. Select either **Windows Authentication** or **SQL Server Authentication**.
4. Select the default database from the **Default database** list.
5. Select the **User Mapping** page.
6. In the **Map** column, check the check box for the database that your login can access.

   By default, the login name appears in the **User** column. Leave this value.
7. In the **Default Schema** column, enter the default schema.

   For use with Silk Test Workbench, the default schema must match the schema of the previously created user with system admin rights.
8. In the **Database role membership for** list, leave the default option **public** selected.
9. Check the **db_datareader** and **db_datawriter** check boxes.
10. Click **OK**.

Important: To use SQL Server Express with multiple users, each SQL Server Express installation must be enabled for remote connectivity. To enable SQL Server Express for remote connectivity, refer to the Microsoft support article *How to enable remote connections on SQL Server* .

Preparing the SQL Server Silk Test Workbench Database

Note: SQL Server Authentication Mode must be set to: SQL Server and Windows (Mixed Mode) to allow connection to the SQL Server database through the Database Maintenance utility. This setting can be changed after any database maintenance tasks are performed.

Note: You can only perform the tasks described in this topic if the SQL Server database is empty.

You must prepare your new SQL Server database for use with Silk Test Workbench. This section describes how to use the Database Maintenance utility to populate the database with Silk Test Workbench tables.

1. Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Administration** > **Database Maintenance** or (in Microsoft Windows 10) **Start** > **Silk** > **Database Maintenance**. The **Database Maintenance** utility starts.
2. Click **File** > **New Database** > **SQL Server**. The **SQL Server Data Source Connection** dialog box opens.
3. Type the name of the new SQL Server data source or click **Browse** and select a data source name from the **Select ODBC Data Source** dialog box.
4. In the appropriate text boxes, type the name of the database owner, which is the user with the role db_owner, and the SQL Server user ID and password, and then click **Create**.

   Note: Only one schema can be associated with a DSN name (ODBC connection). If the chosen DSN name is already associated with a schema, click **Replace** to change the association of the DSN name from its current schema to the new schema (users will no longer be able to access Silk Test Workbench tables in the old schema), or click **Alias** to create an alias for the DSN name.

You are now ready to start Silk Test Workbench and log on.

Configuring an Oracle Database

This section describes how to set up an Oracle database with Silk Test Workbench.

Note: The following instructions assume familiarity with the Oracle database management system. We recommend that a database administrator or a person with general knowledge of database administration performs database setup and configuration.

Each client computer requires an ODBC data source to connect to the database. As a result, connecting to an Oracle database for Silk Test Workbench has the following requirements:

- Each computer running Silk Test Workbench must be able to access the computer where the database resides and have a new data source configured.
- Silk Test Workbench must be installed on all computers that access the database.
- The appropriate database client connectivity software must be installed on any computer that runs Silk Test Workbench.
- The database connection must be configured for use with Silk Test Workbench.

Selecting an Authentication Method

Silk Test Workbench supports the use of either Oracle OS authentication (Windows NT) or Oracle database authentication. There are additional installation procedures required when using Oracle OS authentication.

Oracle OS Authentication

This topic provides instructions for creating users and granting connection rights when using Oracle OS authentication. Prior to performing these steps, consider the following:

- You must be a member of the `ORA_DBA` group and have `SYSDBA` privileges.
- Users must be created in such a way that Oracle knows they are authenticated via a DOMAIN controller.
- Use SQL*Plus or SQL*Plus Worksheet to create users. If you are not familiar with SQL*Plus, consult your Oracle documentation.
- Users must be created within a Windows NT domain and added to the `ORA_DBA` group. Refer to your Microsoft Windows documentation for more information about using a Windows domain.
- When setting up users for Silk Test Workbench in Oracle, create a schema to house the actual tables in addition to the remotely authenticated users. The schema is not domain authenticated, but is password authenticated.

1. Start SQL*Plus or SQL*Plus Worksheet.
2. Log on as SYSTEM and connect as SYSDBA.
3. Type commands to create each user as "DOMAIN\USERNAME" IDENTIFIED EXTERNALLY, and grant connection rights and system privileges as shown in the following example.

   For each user as "DOMAIN\USERNAME". Type DOMAIN\USERNAME in uppercase and between double quotation marks, where DOMAIN\USERNAME is each user's domain and user name.

   For example:

```
CREATE USER "NT-DOMAIN\JEFF" IDENTIFIED EXTERNALLY;
GRANT SELECT ANY SEQUENCE TO "NT-DOMAIN\JEFF";
GRANT UNLIMITED TABLESPACE TO "NT-DOMAIN\JEFF";
GRANT "CONNECT" TO "NT-DOMAIN\JEFF";
GRANT "RESOURCE" TO "NT-DOMAIN\JEFF";
GRANT "SELECT_CATALOG_ROLE" TO "NT-DOMAIN\JEFF";
```

   Important: Microsoft operating systems return the DOMAIN\USERNAME when the user is queried. If this does not match what is defined in the database, Oracle OS authentication will fail with ORA-1017.
4. Test that Oracle OS authentication is set up correctly by connecting to the database via SQL*Plus.
   a) Type a "/" for the user name.
   b) Leave the Password text box empty, and enter the host string appropriately.
   c) Click **OK**. If you can log on, Oracle OS authentication is working properly.

Oracle Database Authentication

This topic provides instructions for creating users and granting connection rights when using Oracle database authentication. Prior to performing these steps, consider the following:

- You must be a member of the ORA_DBA group and have SYSDBA privileges.
- Use SQL*Plus or SQL*Plus Worksheet to create users. If you are not familiar with SQL*Plus, consult your Oracle documentation.
- When setting up users for Silk Test Workbench in Oracle, create a schema to house the actual tables in addition to the remotely authenticated users. The schema is not domain authenticated, but is password authenticated.

1. Start SQL*Plus or SQL*Plus Worksheet.
2. Log on as SYSTEM and connect as SYSDBA.
3. Type commands to create each user as "USERNAME" IDENTIFIED BY "ORACLEPASSWORD", and grant connection rights and system privileges as shown in the following example.

   For each user as "USERNAME". Type USERNAME in uppercase and between double quotation marks, where DOMAIN\USERNAME is each user's user name.

   For example:

```
CREATE USER "JEFF" IDENTIFIED BY "ORACLEPASSWORD";
GRANT SELECT ANY SEQUENCE TO "JEFF";
GRANT UNLIMITED TABLESPACE TO "JEFF";
GRANT "CONNECT" TO "JEFF";
GRANT "RESOURCE" TO "JEFF";
GRANT "SELECT_CATALOG_ROLE" TO "JEFF";
```

Creating a New Oracle Database

Create a new Oracle database with the following character set:

- UTF8 with Oracle versions prior to Oracle 12.
- AL32UTF8 with Oracle 12 or later.

For more information, refer to the Oracle product documentation.

Setting Up Oracle OS Authentication

Note: This procedure is only necessary if you use Oracle OS authentication. If you use Oracle database authentication, continue by setting up the Oracle client.

1. On the server where the database is located, locate the Oracle Initialization file in the directory where Oracle was installed.
2. In the Oracle Initialization file, set values for the following parameters:
   - `remote_login_passwordfile = none`
   - `remote_os_authent = true`
   - `os_authent_prefix = ""`

   Note: If any of the parameters and values do not exist, you must add them. In Oracle 11g, the `remote_os_authent` and `os_authent_prefix` parameters are set by default.
3. Locate the `SQLNET.ORA` file in the directory where Oracle was installed.

   The file is located in `ORACLE_HOME\Network\Admin` directory under the main Oracle installation directory where `ORACLE_HOME` is the name assigned for the Oracle home during installation.
4. Open `SQLNET.ORA` and set the value of the following parameter as follows:

   ```
   sqlnet.authentication_services = (NTS)
   ```

   Note: Ensure this parameter is not commented out with a # at the beginning of the line. You may need to add this parameter.
5. Access the **Run** dialog box from the **Start** menu.
6. In the **Open** text box, type `regedit` and then click **OK**. The **Registry Editor** dialog box opens.
7. In the path `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOMEn` (where *n* is the number associated with the Oracle installation), right-click `HOMEn` and choose **New** > **String Value**. A new string value appears in the right pane of the registry editor.
8. Replace the default name, `New Value #1` with `OSAUTH_PREFIX_DOMAIN`.
9. Double-click the string value you just created. The **Edit String** dialog box opens.
10. In the **Value data** text box, type `TRUE` and then click **OK**.

Setting Up the Oracle Client

On the client computer from which you will connect:

1. Locate the `SQLNET.ORA` file in the directory where Oracle was installed.

   The file is located in `ORACLE_HOME\Network\Admin` directory under the main Oracle installation directory where `ORACLE_HOME` is the name assigned for the Oracle home during installation.
2. Open `SQLNET.ORA` and set the value of the following parameter as follows:

   ```
   sqlnet.authentication_services = (NTS)
   ```

   Note: Ensure this parameter is not commented out with a # at the beginning of the line. You may need to add this parameter.
3. Open the `TNSNAMES.ORA` file in the directory where Oracle was installed.

   This file is located in `ORACLE_HOME\Network\Admin` directory.
4. Copy the server's database entry to the `TNSNAMES.ORA` file on the client computer.
   For example:

   ```
   ORACLE_SID.DOMAIN.COM=
   (DESCRIPTION=
   (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=TCP)(HOST=SERVERNAME)(PORT=1521))
   ```

```
)
(CONNECT_DATA=
(SERVICE_NAME=ORACLE_SID)
)
)
```

Preparing the Oracle Database for Client Connectivity

Note: This procedure is only necessary if you are using Oracle OS authentication and an Oracle client to set up the Silk Test Workbench database. If you are using Oracle database authentication, you can begin to set up users.

1. On the computer that has User Manager for Domains, access the Windows **Control Panel**, and then choose **Administrative Tools** > **Computer Management**

   The **Computer Management** utility opens.
2. In the tree view, click **Local Users and Groups**.
3. Click **Groups**.

   If Oracle is installed, you should see a group named `ORA_DBA`. If not, you must add it.

Next, create server access for users.

Creating the ORA_DBA Group

1. In the tree view of the **Computer Management** utility, right-click **Groups** and choose **New Group**. The **New Group** dialog box opens.
2. In the **Group Name** text box, type `ORA_DBA`.
3. Click **Create**.
4. Click **Close**.

Creating Server Access for Users

To create individual user accounts:

1. In the tree view of the **Computer Management** utility, click **Local Users and Groups**.
2. Click **Groups**.
3. In the right pane, double-click **ORA_DBA**. The **ORA_DBA Properties** dialog box opens.
4. Click **Add**. The **Select Users, Computers, or Groups** dialog box opens.
5. Click **Advanced**, then use the **Common Queries** tab to search for users, click **Find Now** and select users from the list.

   Alternatively, click in the **Enter the object names to select** text box and type the following: `domain name\user ID`

   Where `domain name` is the name of the domain where the user is located and `user ID` is the network ID of the user you want to give access to.
6. Click **OK** to save the user information and return to the **ORA_DBA Properties** dialog box.

   Repeat steps 4–6 to create any additional user accounts.
7. Click **OK** in the **Users Properties** dialog box to return to the **Computer Management** utility.

Setting Up Users for Silk Test Workbench in Oracle

This section provides instructions for creating users and granting connection rights for users. This procedure is necessary when using either Oracle OS authentication or Oracle database authentication. Separate instructions are provided for each type of authentication.

Oracle OS Authentication

Users must be created in such a way that Oracle knows they are authenticated via a DOMAIN controller.

Important:

When setting up users for Silk Test Workbench in Oracle, a schema must be also created to house the actual tables in addition to the remotely authenticated users. The schema is not domain authenticated, but is password authenticated. For more information, see *Creating the Password Authenticated Schema*.

Users must be created within a Windows NT domain. Refer to your Microsoft Windows documentation for more information about using a Windows domain.

Users must be added to the ORA_DBA group. Use SQL*Plus or SQL*Plus Worksheet to create users. If you are not familiar with SQL*Plus, consult your Oracle documentation.

The user performing these steps must be a member of the ORA_DBA group and have SYSDBA privileges. To specify users that can access the Oracle database:

1. Start SQL*Plus or SQL*Plus Worksheet.
2. Log on as `SYSTEM` and connect as `SYSDBA`.
3. Type commands to create each user as `"DOMAIN\USERNAME" IDENTIFIED EXTERNALLY`, and grant connection rights for each user as `"DOMAIN\USERNAME"`. Type `DOMAIN\USERNAME` in uppercase and between double quotation marks, where `DOMAIN\USERNAME` is each user's domain and user name. For example:

   ```
   SQL> create user "NT-DOMAIN\JEFF" IDENTIFIED EXTERNALLY;
   SQL> grant connect, resource to "NT-DOMAIN\JEFF";
   ```

   Note: `SELECT_ANY_SEQUENCE` and `SELECT_CATALOG_ROLE` must also be granted to the user. For example:

   ```
   SQL> grant select any sequence to "JEFF";
   SQL> grant select_catalog_role to "JEFF";
   ```

   Microsoft operating systems return the `DOMAIN\USERNAME` when the user is queried. If this does not match what is defined in the database, NT authentication will fail with ORA-1017.
4. Test that Windows NT authentication is set up correctly by connecting to the database via SQL*Plus. Type a `"/"` for the user name. Leave the **Password** field blank, and enter the host string appropriately. Click **OK**. If you are able to log on, Windows NT authentication is working properly.

Oracle Database Authentication

Important: When setting up users for Silk Test Workbench in Oracle, a schema must also be created to house the actual tables in addition to the remotely authenticated users. The schema is password authenticated. For more information, see *Creating the Password Authenticated Schema*.

Use SQL*Plus to create users. If you are not familiar with SQL*Plus, consult your Oracle documentation. The user performing these steps must be a member of the ORA_DBA group and have SYSDBA privileges.

To specify users that can access the Oracle database:

1. Start SQL*Plus or SQL*Plus Worksheet.
2. Log on as `SYSTEM` and connect as `SYSDBA`.
3. Type commands to create each user as `"USERNAME" IDENTIFIED BY "ORACLEPASSWORD"`, and grant connection rights for each user as `"USERNAME"`. Type `USERNAME` in uppercase and between double quotation marks, where `DOMAIN\USERNAME` is each user's user name. For example:

   ```
   SQL> create user "JEFF" IDENTIFIED BY "ORACLEPASSWORD";
   SQL> grant connect, resource to "JEFF";
   ```

Note: SELECT_ANY_SEQUENCE, SELECT_CATALOG_ROLE, and SELECT_ANY_DICTIONARY must also be granted to the user. For example:

```
SQL> grant select any sequence to "JEFF";
SQL> grant select_catalog_role to "JEFF";
SQL> grant select any dictionary to "JEFF";
```

Creating the Password Authenticated Schema

To set up the password authenticated schema into which the Silk Test Workbench tables are stored, perform the following steps:

Note: The password authenticated schema should not be more than five characters in length because of Oracle character limitations in SQL series.

1. Start SQL*Plus or SQL*Plus Worksheet.
2. Log on as SYSTEM and connect as SYSDBA.
3. Type commands to create each user as "USERNAME" IDENTIFIED BY "ORACLEPASSWORD", and grant connection rights for each user as "USERNAME".

   Type USERNAME in uppercase and between double quotation marks, where USERNAME is each user's user name. For example:

   ```
   SQL> create user "TOM" IDENTIFIED BY "ORACLEPASSWORD";
   SQL> grant connect, resource to "TOM";
   ```

Preparing the Oracle Database

You must prepare your new Oracle database for use with Silk Test Workbench. This section describes how to use the **Database Maintenance** utility to populate the database with Silk Test Workbench tables.

1. Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Administration** > **Database Maintenance** or (in Microsoft Windows 10) **Start** > **Silk** > **Database Maintenance**. The **Database Maintenance** utility starts.
2. Click **File** > **New Database** > **Oracle**. The **Oracle Data Source Connection** dialog box opens.
3. Type the name of the new Oracle data source or click **Browse** and select a data source name from the **Select ODBC Data Source** dialog box.
4. In the appropriate boxes, type the name of the Oracle user ID, and password, and then click **Create**. The **Schema** box is filled in automatically with the same information as the User ID.

You are now ready to start Silk Test Workbench and log on.

Setting Up a SQL Server or Oracle Database Without a Domain

We strongly recommend that you use authentication within a domain when setting up an Oracle or SQL Server database with Silk Test Workbench. However, if you are not on a domain, it is still possible to set up these databases.

This section provides specific instructions for setting up a SQL Server or Oracle database with a domain.

Creating Users Without a Domain

To set up a SQL Server or Oracle database without a domain, you must first create a user account on both the client and server computers.

Note: This procedure only applies when using Windows NT authentication.

1. Access the Windows **Control Panel**, and then choose **Administrative Tools** > **Computer Management**

   The **Computer Management** utility opens.
2. In the left pane, click **Local Users and Groups**.
3. Under **Local Users and Groups**, right-click the **Users** folder and choose **New User**. The **New User** dialog box opens.
4. Type a valid user name in the **User name** text box, and the corresponding user's full name in the **Full Name** text box.
5. Type a valid password in both the **Password** and **Confirm password** text boxes, and then click **Create**.
6. Repeat these steps on each client computer using the username and password.

Setting Up SQL Server Without a Domain

To set up SQL server without a domain:

1. Access the Windows **Control Panel**, and then choose **Administrative Tools** > **Computer Management**

   The **Computer Management** utility opens.
2. In the left pane, expand **Local Users and Groups**, and then click **Groups**.
3. In the right pane, double-click **Users**.

   The **Users Properties** dialog box opens.
4. Click **Add**.

   The **Select Users, Computers, or Groups** dialog box opens.
5. Click **Advanced**, then use the **Common Queries** tab to search for users.
6. Click **Find Now**, and then select users from the list.

   Alternatively, click in the **Enter the object names to select** text box and enter the `[user ID]` of the user you want to give access.`[user ID],`:
7. Click **OK** to save the user information and return to the **Users Properties** dialog box.
8. Repeat the previous steps to create additional user accounts.
9. Click **OK**.

Setting Up an Oracle Database Without a Domain

To set up Oracle without a domain, follow the procedure for creating server access for individual user accounts; however, substitute the user name you created in the procedure for creating users without a domain.

1. In the **Enter the object names to select** text box, type the user name instead of `[domain name]\[user ID]` or `[domain name]\[group name]`.

   The **Computer Management** utility appears.
2. Set up users.

   When using SQL*Plus to create users, substitute `[computer name\user name]` for `[domain name\user name]` as shown in the following example.

```
SQL> create user "FH0002RED\USER1" IDENTIFIED EXTERNALLY;
SQL> grant connect, resource to "FH0002RED\USER1";
```

   Note:

   For the `[computer name\user name]`, use uppercase and enclose in quotes.

Creating a Data Source Name

To work with a database that you have set up for Silk Test Workbench, use the ODBC utility to create a data source name (DSN) for the Silk Test Workbench database on each machine that connects to the database.

Note: This step does not apply to default Access databases installed by the Silk Test Workbench installer. For these databases, a data source name is automatically created.

Creating a Data Source Name for an Access Database

On each client computer connecting to an Access database, perform the following steps to set up a DSN:

1. Navigate (in Microsoft Windows 7) to **Start** > **Silk** > **Silk Test** > **Administration** > **Data Sources (ODBC)** or (in Microsoft Windows 10) to **Start** > **Windows Administrative Tools** > **ODBC Data Sources (32-bit)**. The **ODBC Data Source Administrator** dialog box opens.
2. Click the **System DSN** tab, then click **Add**. The **Create New Data Source** dialog box opens. This dialog box lists the available ODBC drivers.
3. From the list of drivers, select the **Microsoft Access Driver** and click **Finish**. The **ODBC Microsoft Access Setup** dialog box opens.
4. In the **Data Source Name** text box, type the data source name.

   Enter a logical name as this name appears as the database name in the logon dialog box.
5. In the **Description** text box, type a description for the data source.
   For example, `Connection to Silk Test Workbench database`.
6. Click **Select** and browse for your Access database.
7. On the **ODBC Microsoft Access Setup** dialog box, click **OK**.
8. On the **ODBC Administrator** dialog box, click **OK** to complete the process.

Creating a Data Source Name for a SQL Server Database

Note: Silk Test Workbench cannot use 64-bit DSNs. To create a DSN for a 64-bit machine, click (in Microsoft Windows 7) **Start** > **Silk** > **Silk Test** > **Administration** > **Data Sources (ODBC)** or (in Microsoft Windows 10) **Start** > **Windows Administrative Tools** > **ODBC Data Sources (32-bit)** and create a 32-bit DSN. You can also use the WOW64 tools located at `C:\WINDOWS\SysWOW64\odbcad32.exe`.

On each client computer connecting to a SQL server database, perform the following steps to set up a DSN:

1. Navigate (in Microsoft Windows 7) to **Start** > **Silk** > **Silk Test** > **Administration** > **Data Sources (ODBC)** or (in Microsoft Windows 10) to **Start** > **Windows Administrative Tools** > **ODBC Data Sources (32-bit)**. The **ODBC Data Source Administrator** dialog box opens.
2. Click the **System DSN** tab, then click **Add**. The **Create New Data Source** dialog box opens. This dialog box lists the available ODBC drivers.
3. From the list of drivers, select the **ODBC Driver 11 for SQL Server** and click **Finish**. The **Create a New Data Source to SQL Server** dialog box opens.
4. In the **Name** box, type the data source name.

   Enter a logical name as this name appears as the database name in the logon screen.
5. In the **Description** text box, type a description for the data source.
   For example, `Connection to Silk Test Workbench database`.
6. In the **Server** text box, enter the server name or select it from the drop-down list.
7. Click **Next**.

8. Perform one of the following:
   - If user(s) using the DSN to access the SQL Server database are connecting with non-native credentials, select the **With Windows authentication** using the network login ID option.
   - If user(s) using the DSN to access the SQL Server database are connecting with native authentication, select the **With SQL Server authentication using a login ID and password entered by the user** option. Make sure the **Connect to SQL Server to obtain the default settings for the additional configuration options** check box is checked. Type the users native SQL Server logon credentials in the **Login ID** and **Password** text boxes.
9. Click **Next**.
10. Check the **Change the default database to** check box, and then select your database name from the list.
11. Accept all the other entries and click **Next** until the last panel in the wizard appears.
12. Accept the defaults and click **Finish**.
13. To test the connection, click **Test Data Source**. The **SQL Server ODBC Data Source Test** dialog box opens.
14. Click **OK**.

    The **ODBC Data Source Administrator** dialog box reappears. The newly created data source appears in the **System Data Sources** list.
15. Click **OK**.

Creating a Data Source Name for an Oracle Database

Note: Silk Test Workbench cannot use 64-bit DSNs. To create a DSN for a 64-bit machine, click (in Microsoft Windows 7) **Start** > **Silk** > **Silk Test** > **Administration** > **Data Sources (ODBC)** or (in Microsoft Windows 10) **Start** > **Windows Administrative Tools** > **ODBC Data Sources (32-bit)** and create a 32-bit DSN. You can also use the WOW64 tools located at `C:\WINDOWS\SysWOW64\odbcad32.exe`.

On each client computer connecting to an Oracle database, perform the following steps to set up a DSN:

1. Navigate (in Microsoft Windows 7) to **Start** > **Silk** > **Silk Test** > **Administration** > **Data Sources (ODBC)** or (in Microsoft Windows 10) to **Start** > **Windows Administrative Tools** > **ODBC Data Sources (32-bit)**. The **ODBC Data Source Administrator** dialog box opens.
2. Click the **System DSN** tab, then click **Add**. The **Create New Data Source** dialog box opens. This dialog box lists the available ODBC drivers.
3. From the list of drivers, select the appropriate Oracle ODBC driver for the version of Oracle database being used (not the Microsoft ODBC for Oracle driver), and then click **Finish**. The **Oracle ODBC Driver Configuration** dialog box opens.
4. In the **Oracle Data Source Name** text box, type the data source name.

   Enter a logical name as this name appears as the database name in the logon dialog box.
5. In the **Description** text box, type a description for the data source.
   For example, `Connection to Silk Test Workbench database`.
6. In the **TNS Service Name** text box, select the service name for the database you want to connect to in the format `ORACLE_TNS.DOMAIN`, where `ORACLE_TNS` is usually the Oracle SID that was assigned to the database upon creation and DOMAIN is the domain to which you are connecting. The domain is usually optional.

   Note: To connect Silk Test Workbench, the TNS service name is mandatory.
7. If using Oracle authentication, type a valid user ID for the database being connected to in the **User ID** text box. Otherwise, leave the **User ID** text box empty.
8. To test the connection, click **Test Connection**.

Note: When using Oracle database authentication, type a username, and then a password. For Oracle OS authentication, do not specify a username.

A message box appears stating the connection was successful.

9. Click **OK**.

Connecting to a Database

After installing Silk Test Workbench and configuring a database, you must create a database connection. This connection defines the relationship between a Silk Test Workbench database and the local Silk Test Workbench installation. This section contains topics describing how to log on and create a database connection.

Logging On

1. Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Clients** > **Silk Test Workbench** or (in Microsoft Windows 10) **Start** > **Silk** > **Silk Test Workbench**. Silk Test Workbench starts and displays the **Login to SilkTest Workbench** dialog box.
2. From the **Authentication** list box, select if you want to use the **Silk Test Workbench** authentication, or the **Windows** authentication.

   Note: You can use the **Windows** authentication only for users that were added to the Silk Test Workbench users by using **Tools** > **Administration**.
3. If you are using the **Silk Test Workbench** authentication, type a user name in the **User Name** field and a password in the **Password** field. If you are using the **Windows** authentication, you do not have to provide a user name and password.

   Note: If you are using Silk Test Workbench for the first time, an administrative user name and password may be required. The default user name is `Admin` and the default password is `admin`. Change this password after logging on to prevent unauthorized access. You can change your logon password at any time.
4. Select the database to use from the **Database** list.

   For a database to appear in the **Database** list, you must first configure its database connection for use. To configure a database connection, click **Configure** on the **Logon** dialog box.
5. Click **OK**.

   If you are the first user that is logging in to this database, Silk Test Workbench will create a default project named *Project1* in the database. All assets that you create will be stored by default into this project. You can change the project name by clicking **Edit Project**.

   The **Start Screen** opens. If you are logging in to Silk Test Workbench for the first time, Silk Test Workbench selects *Project1* or the first available non-global object as the active object.

Changing Your Logon Password

Use the following procedure to change your logon password.

1. Click **Tools** > **Change Password**. The **Change Password** dialog box opens. The logon name of the current user displays in the title bar of the dialog box.
2. Change your password and then click **OK**. Changes take effect the next time the User ID is used to log on.

Configuring a Database Connection

Before you can use a database with Silk Test Workbench, you must configure its database connection. The configuration process consists of using the **Configure Database Connection** dialog box to perform the following tasks:

- Specify the database connection data required to initiate the database connection.
- Verify the connection to the database.
- Save the database connection data.

All configured database connections appear in the **Database** list that is located on the Silk Test Workbench dialog box.

You can also use the **Configure Database Connection** dialog box to view, edit, and remove any existing configured database connections.

Note: Before configuring a database connection for use with Silk Test Workbench, you must create and configure a database instance and ODBC Data Source Name (DSN).

Note: Silk Test Workbench cannot use 64-bit DSNs. To create a DSN for a 64-bit machine, click (in Microsoft Windows 7) **Start** > **Silk** > **Silk Test** > **Administration** > **Data Sources (ODBC)** or (in Microsoft Windows 10) **Start** > **Windows Administrative Tools** > **ODBC Data Sources (32-bit)** and create a 32-bit DSN. You can also use the WOW64 tools located at `C:\WINDOWS\SysWOW64\odbcad32.exe`.

1. Click **Configure** on the **Login to** Silk Test Workbench dialog box. The **Configure Database Connection** dialog box opens.
2. In the **Data Source Type** section, select the type of data source.
3. From the **Select Data Source** list, select from the list of available DSNs.
4. Check the **Use as a Silk Test Workbench database** check box to add the selected DSN to the list of configured database connections that appear in the **Database** list on the **Login to** Silk Test Workbench dialog box.

   You must check this check box when configuring a database connection for the first time. For existing configured database connections, unchecking the **Use as a Silk Test Workbench database** check box and clicking **Apply** removes the selected DSN from the **Database** list on the dialog box. Silk Test Workbench does not retain the associated database connection data.
5. In the **Database Settings** section, specify the appropriate information for the selected DSN. For Access, the **Database** box is read-only and displays the location of the database file. This value is read directly from the ODBC DSN. The other **Database Settings** text boxes are not applicable for Access and are disabled. For Oracle, the **Server** text box is read-only and displays only the database name. This value is read directly from the ODBC DSN.
6. In the **Database** box, type the appropriate value. For SQL Server/MSDE, the **Database** text box is modifiable and allows you to type the appropriate database name.
7. In the **Owner** box, type the appropriate value.
8. Select a type of authentication.

   Authentication options only apply to Oracle or SQL Server/MSDE databases. You can choose to use either Windows NT authentication or authenticate using the database's native authentication capabilities. When selecting database authentication, you must provide the user name and password in the appropriate text boxes.
9. Click **Validate** to verify that you can connect to the database using the specified database connection data.
10. Click **Apply** to save the database connection data.
11. Click **Close** to dismiss the dialog box.

Database Maintenance

The **Database Maintenance** utility performs several maintenance tasks, such as updating the database schema, unlocking records in the database, and compacting the database.

Before performing maintenance tasks, you must configure your Silk Test Workbench database connection. Additionally, for databases not installed by the Silk Test installer, a System Administrator must create a new

ODBC data source before the **Database Maintenance** utility can be used to create a new Silk Test Workbench ODBC database. For more information, refer to the *Silk Test Installation Guide*.

Opening a Silk Test Workbench Database

You can open a previously created Access, SQL Server, or Oracle Silk Test Workbench database to perform maintenance tasks such as updating the database schema, unlocking records in the database, and compacting the database.

Opening a Microsoft SQL Server Database

Use the following procedure to start the **Database Maintenance** utility and open a Microsoft SQL Server database.

1. Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Administration** > **Database Maintenance** or (in Microsoft Windows 10) **Start** > **Silk** > **Database Maintenance**. The **Database Maintenance** utility starts.
2. Click **File** > **Open Database** > **SQL Server**. The **SQL Server Data Source Connection** dialog box opens.
3. Click **Browse**. The **Select ODBC Data Source** dialog box opens.
4. Select the desired Microsoft SQL Server DSN, and then click **Select**. The **SQL Server Data Source Connection** dialog box opens and displays the selected DSN in the **SQL Server Data Source** text box.

   Note: The DSN for the default Silk Test Microsoft SQL Server 2008 Express database is Silk Test_SQL.
5. In the **Owner**, **User ID**, and **Password** text boxes, enter the appropriate information for the Silk Test Microsoft SQL Server database instance.

   Note: The owner for the default Silk Test Microsoft SQL Server 2008 Express database is Silk TestDB, and the user with admin rights is `SilkTestAdmin` with password `SilkTestAdmin`.
6. Click **Open**. The database opens and the **Database Maintenance** utility's **Tools** menu options become available. If there are users currently connected to the selected database, a message is displayed. For more information, see *Performing Maintenance on a Database with Connected Users*.

Opening an Oracle Database

Use the following procedure to start the **Database Maintenance** utility and open an Oracle database.

1. Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Administration** > **Database Maintenance** or (in Microsoft Windows 10) **Start** > **Silk** > **Database Maintenance**. The **Database Maintenance** utility starts.
2. Click **File** > **Open Database** > **Oracle**.

   The **Oracle Data Source Connection** dialog box opens.
3. Click **Browse**. The **Select ODBC Data Source** dialog box opens.
4. Select the desired Oracle DSN, and then click **Select**. The **Oracle Data Source Connection** dialog box opens and displays the selected DSN in the **Oracle Data Source Name** text box.
5. In the **Schema**, **User ID**, and **Password** text boxes, enter the appropriate information for the Silk Test Oracle database instance.
6. Click **Open**. The database opens and the **Database Maintenance** utility's **Tools** menu options become available. If there are users currently connected to the selected database, Silk Test Workbench displays a message.

For more information, see *Performing Maintenance on a Database with Connected Users*.

Opening an Access Database

Use the following procedure to start the **Database Maintenance** utility and open a Microsoft Access database.

1. Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Administration** > **Database Maintenance** or (in Microsoft Windows 10) **Start** > **Silk** > **Database Maintenance**. The **Database Maintenance** utility starts.
2. Click **File** > **Open Database** > **Access**. The **Access Database Connection** dialog box opens.
3. Click **Browse**. The **Select ODBC Data Source** dialog box opens.
4. Select the desired Access DSN and then click **Select**.
5. Click **Open**.

   For more information, see *Performing Maintenance on a Database with Connected Users*.

   The database opens and the **Database Maintenance** utility's **Tools** menu options become available. If there are users currently connected to the selected database, Silk Test Workbench displays a message.

Updating a Database Version

When upgrading Silk Test Workbench to the current version, the existing database must be converted to be compatible with the new software. Silk Test Workbench automatically checks the database schema version when a database is loaded and displays a warning if database upgrading is required. The warning appears any time the database schema is not the same as the current Silk Test Workbench schema. The database can be converted with the **Database Maintenance** utility.

The conversion speed is related to the amount of data being converted. Databases with large amounts of data may experience long conversion times. After the conversion, the current database version number appears.

Note: When you convert the database to a new schema, you can no longer access the database with an older version of Silk Test Workbench, which means all Silk Test Workbench users will have to upgrade to the new version of Silk Test Workbench.

1. If necessary, cancel the Silk Test Workbench logon request and ensure that users are not logged on to the database.
2. Start the **Database Maintenance** utility and open the database. The **Database Maintenance** utility main window appears.
3. Click **File** > **Open Database** and select the type of database. The **Database Connection** dialog box appears.
4. Type the database name and password or click **Browse**.
5. Click **Open**.

   If the database requires updating, a message box appears informing you to click **Tools** > **Convert** to update the database.
6. Click **OK**. The database opens and the **Database Maintenance** utility's **Tools** menu options are enabled.
7. Click **Convert**. The **Convert Database** dialog box opens and asks if the current database should be backed up prior to conversion. Choose to back up the database.
8. Click **Convert Now** to update a database to the current schema version.
9. Click **Close** when the conversion completes.

Copying Database Records

The **Database Maintenance** utility can make a duplicate copy of a database by copying records from one database to another database. For example, if a site uses Microsoft Access as the Silk Test Workbench

asset repository, and the desire is to store Silk Test Workbench data in an Oracle database, use the **Database Maintenance** utility to copy the data stored in the Access database to the Oracle database. When the copy utility is used, all Silk Test Workbench data is automatically copied to a format that is readable by the target database.

Copy operations not should be performed on network databases. Instead, copy the database to a local workstation, perform the copy function, then copy the database back to the network location. This ensures that network disruptions and interruptions do not interfere with the copy task.

1. Ensure that users are not connected to the database.
2. Start the **Database Maintenance** utility and open the database to be copied.
3. Click the **Copy** toolbar button. The **Copy Destinations** dialog box opens.
4. Select the type of database from which to copy, and then click **OK**.
   - To copy an Access database:
     1. In the **DSN** text box, click **Browse** and select the DSN of the database to copy. The path and name of the database appears in the **Access** text box.
     2. Click **Copy**. The **Copy Table Status** dialog box opens and displays the progress of the database copy operation.
   - To copy a Microsoft SQL Server database:
     1. Type the name of the Microsoft SQL Server data source or click **Browse** to select a data source name. The owner name defaults to dbo.
     2. Type the user ID. The user must have dbo privileges.
     3. Type the password for the Silk Test Workbench database instance on the database server. This User ID and password is used to attempt a connection to the Silk Test Workbench database instance on the database server.
     4. Click **Copy**. If the owner name, user ID, and password are accepted by the database server, the **Data Source Creation Status** dialog box opens and displays the progress of the database copy operation.
   - To copy an Oracle database:
     1. Type the name of the new Oracle data source or click **Browse** to select an existing data source name. The **Schema** and **User ID** text boxes have a default of **System**, therefore only a system administrator or someone with knowledge of the appropriate password can perform maintenance if the default settings are used.
     2. Click **Copy**. If the schema name, user ID, and password are accepted by the database server, the **Data Source Creation Status** dialog box opens and displays the progress of the copy operation. The **Database Maintenance** utility begins populating the Silk Test Workbench database instance with the necessary Silk Test Workbench tables.

Unlocking Database Records

When an asset is open, its records are locked to prevent other users from editing the same asset simultaneously. When saving the resource or closing the system, the locks are removed. If you experience a system crash, it is possible that records that were in use remain locked on startup. Silk Test Workbench attempts to unlock records that were inadvertently locked, and originally intended to remain unlocked, after a system crash.

Ensure that all users are logged off the system before using the program. If there are users connected, a message appears.

1. Start the **Database Maintenance** utility and open the database.
2. Click **Unlock Records**. If there are users connected to the database a message similar to the following appears: `Database Maintenance found connected users...`
3. Click **OK**. The **Connected Users** dialog box opens. The machine names of current connected users display in the **Users** list.

4. Click **Send** to send a notice advising these users to log off. Once they are logged off, repeat steps 1 and 2 to unlock the database records.

Note: To send a message to connected users, you must enable the Messenger service on your operating system. To do so, click **Control Panel** > **Administrative Tools** > **Services**, and then select **Messenger** and start the service.

Compacting an Access Database

To ensure optimum performance, regularly compact the database if the database is used on a daily basis. Over time, as resources are updated or deleted, the database file becomes fragmented. Fragmentation wastes disk space and can impair performance. An Access database does not register a decrease in file size when assets are deleted from it until this procedure is performed. The **Database Maintenance** utility lets you compact a Microsoft Access database, removing unwanted records and rewriting the data to a contiguous area of the disk.

Database compact operations should not be performed on network databases. Instead, copy the database to a local workstation, compact the database, then copy it back to the network location. This ensures that network disruptions and interruptions do not interfere with the database maintenance task.

1. Ensure that all users are logged off the system and close Silk Test Workbench.
2. Start the **Database Maintenance** utility and open the database.
3. Click **Compact**. A confirmation dialog box opens.
4. Click **Yes** to proceed. When complete, a dialog box indicates that the database has been successfully compacted.
5. Click **OK**. The database is now available for use.

Configuring Database Maintenance Settings

The **Database Maintenance** utility lets you configure settings that are used during compact, convert, and copy operations.

The backup settings in the **Configure operations for maintenance system** dialog box apply to Microsoft Access databases only. To configure backup settings for other types of databases used as a Silk Test Workbench database, refer to the product documentation of the specific database.

When a database is compacted or converted, the current version can be backed up. Use the **Configure operations for maintenance system** dialog box to specify the maximum number of backup copies to retain. If you specify that you want to retain three compact or convert backups, each time you convert or compact a database, one new backup is created. After three backups are created, the next time you convert or compact, the oldest backup is deleted to enable a new backup to be added.

Select the number of records to buffer during a transaction. A transaction is the process of writing a large amount of data to the database in cycles to improve performance. Each database operation is buffered, until the transaction is committed to the database. However, if any operation fails, then all operations in that transaction fail. You can specify the number of operations to buffer before saving the data to the database.

1. Ensure that all users are logged off the system and close Silk Test Workbench.
2. Start the **Database Maintenance** utility and open the database.
3. Choose **Options** > **Configure**. The **Configure operations for maintenance system** dialog box opens.
4. Double-click the options in the **Settings** tab to specify the number of backups to retain from compacts and conversions, to set the number of records to commit, and to specify the number of seconds before a query attempt fails.

Performing Maintenance on a Database With Connected Users

The **Database Maintenance** utility displays a message if users are connected to the selected database, and enables notification of these users through the **Connected Users** dialog box, which displays connected users in a list. Notify the users to log off the database by using either the default message or creating a unique message, and clicking **Send**.

Avoid performing maintenance on a database that has connected users, because it may result in system instability on the local machine. If the choice is to wait, use the **Refresh List** button to get a current view of connected users.

Note: To receive messages, all users must enable the Windows messenger functionality. To do so, the user must go into Services on their client machine and turn on the Messenger service. The Windows mechanism used to send messages may occasionally send multiple messages.

Running a Database Integrity Check

To successfully run a database integrity check, you require writing permissions on the Silk Test installation directory. During the database integrity check, Silk Test Workbench writes the results of the database integrity check into this directory. If you do not have the writing permissions, your database might get locked for maintenance.

Use the **Database Maintenance** utility to verify the integrity of a database, and to correct errors it may encounter.

1. Open the database, and choose **Tools** > **Run Integrity Check**. The **Verifying Assets** message box appears and give the status of the integrity check. If no errors are found, the message box closes. If an error is encountered, a message box appears prompting you to view the results.
2. Click **Yes** to view the results of the integrity check. The results display in your default text editor. The file containing the results of the integrity checks is saved in a file named `IntegrityCheck.log` in the Silk Test logs directory, which is typically `%LOCALAPPDATA%\Silk\Silk Test\logs`.

Limiting Database Growth

During the recording and playing back of a visual test, Silk Test Workbench captures and stores the screens and controls of the test application at each stage of the recording and playback. Due to the potential of capturing a large number of images and controls, you may experience a high growth rate of the size of your database.

To limit database growth, disable the **Screen capture** and **Control capture** recording and playback settings. For more information, see *Setting Record Output Options* and *Setting Playback Result Visual Test Options*.

Note: Capturing controls allows you to update controls from the **Screen Preview** and insert controls from the test application in a recorded visual test. Additionally, disabling this feature hides the highlight box in the **Screen Preview** that indicates the control associated with the selected step.

As an example, the size of a typical test application's playback result when exported in XML format using the various capture settings is as follows:

- 7.8 MB when capturing both screens and controls
- 5.9 MB when capturing only screens
- 382 KB with screen and control capture disabled

Managing Projects

A project is a collection of assets with assigned access privileges for each user on each project. One of your tasks as an administrator of the Silk Test Workbench environment is to maintain the list of projects. Use projects to store assets in a logical association. For example, all assets associated with a specific application or build can be stored together in a project.

Administrators can create and add projects to the Silk Test Workbench database. When creating a project, the access rights of the creator to the new project reflect the default access rights that are granted to the creator. Storing assets in separate projects provides different levels of security. For example, a user might have full access for *Project1*, but read-only access for *Project2*, and no access for *Project3*.

Only the assets in the projects for which the current user has access to can be viewed. Assets are viewed and managed using the **Asset Browser**.

Note: To optimally use the functionality that Silk Test Workbench provides, create an individual project for each application that you want to test, except when testing multiple applications in the same test.

Common Project

The default project is the Common project, which serves as a central repository for assets used in multiple projects. Assets in the Common project are available to all other projects. The Common project is the active project by default.

Active Project

Newly created assets are added to the project that is set as the Active project. Assets added to the Active project are available only in that project and in all projects that reference the Active project.

Note: Assets in a project can only access other assets in the same project or in a referenced project.

Adding a Project

To add projects to a Silk Test Workbench database, you must log in as an administrator. When creating a project, your access rights to the new project reflect your default access rights.

1. Choose **Tools** > **Administration**.
   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.
   The **Administration** dialog box opens.
2. Click the **Projects** tab.
3. Click **Add**. The **Add Project** dialog box opens.
4. Enter the name of the new project in the **Name** text box.
5. Click **OK**.

Referencing Projects

Reference projects to better organize your assets by reusing code and assets from the referenced project. Additionally, referencing projects enables larger teams to have multiple places to share assets, for example where different teams are testing different areas of an application.

To reference a project from another project:

1. In the Silk Test Workbench menu, select **Tools** > **Administration**. The **Administration** dialog box opens.

2. Select the **Projects** tab.
3. Select the project from which you want to reference another project.
   Projects to which the current user has no access are indicated with a strike through.
4. Click **Modify**.
5. Select the project that you want to reference from the **Available** list.
6. Use the arrow to move the project to the **Referenced** list.

All assets in the referenced project are now also available in the current project.

Viewing Project References

View project references to see which projects are referenced by the current project and which projects reference the current project.

1. In the Silk Test Workbench menu, choose **View** > **Project References**.
2. The project references opens.
3. Click on the **References** tab to see which projects are referenced by the current project.
4. Click on the **Referenced By** tab to see which projects reference the current project.

Defining a Global Project

To reference projects from all other projects in the database:

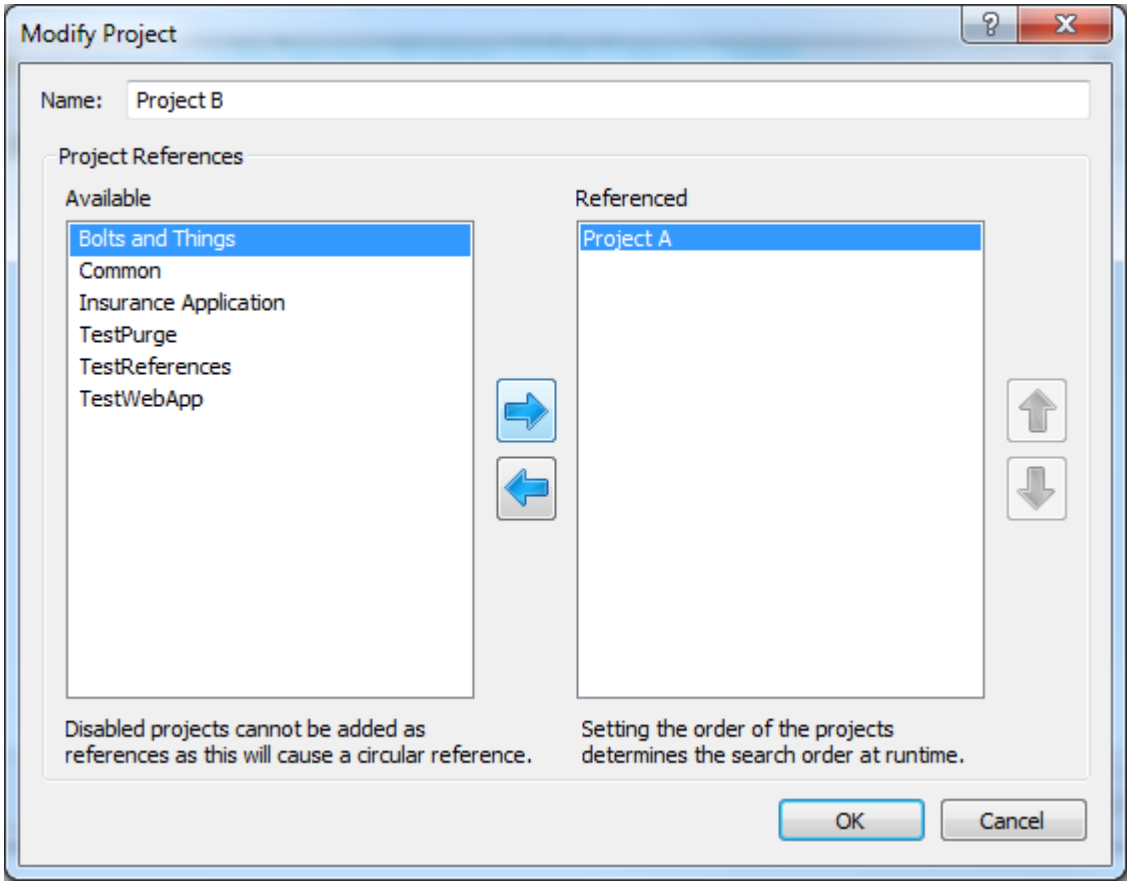
1. In the Silk Test Workbench menu, select **Tools** > **Administration**. The **Administration** dialog box opens.
2. Select the **Projects** tab.
3. Click **Define Global Projects**. The **Define Global Projects** dialog appears. All projects that do not reference any other project and that are not referenced by any other project are listed in the **Available** list.
4. Select the projects that you want to globally reference from the **Available** list.
5. Use the arrow to move the projects to the **Global Projects** list.

All assets in the global project are now also available to all other projects in your database.

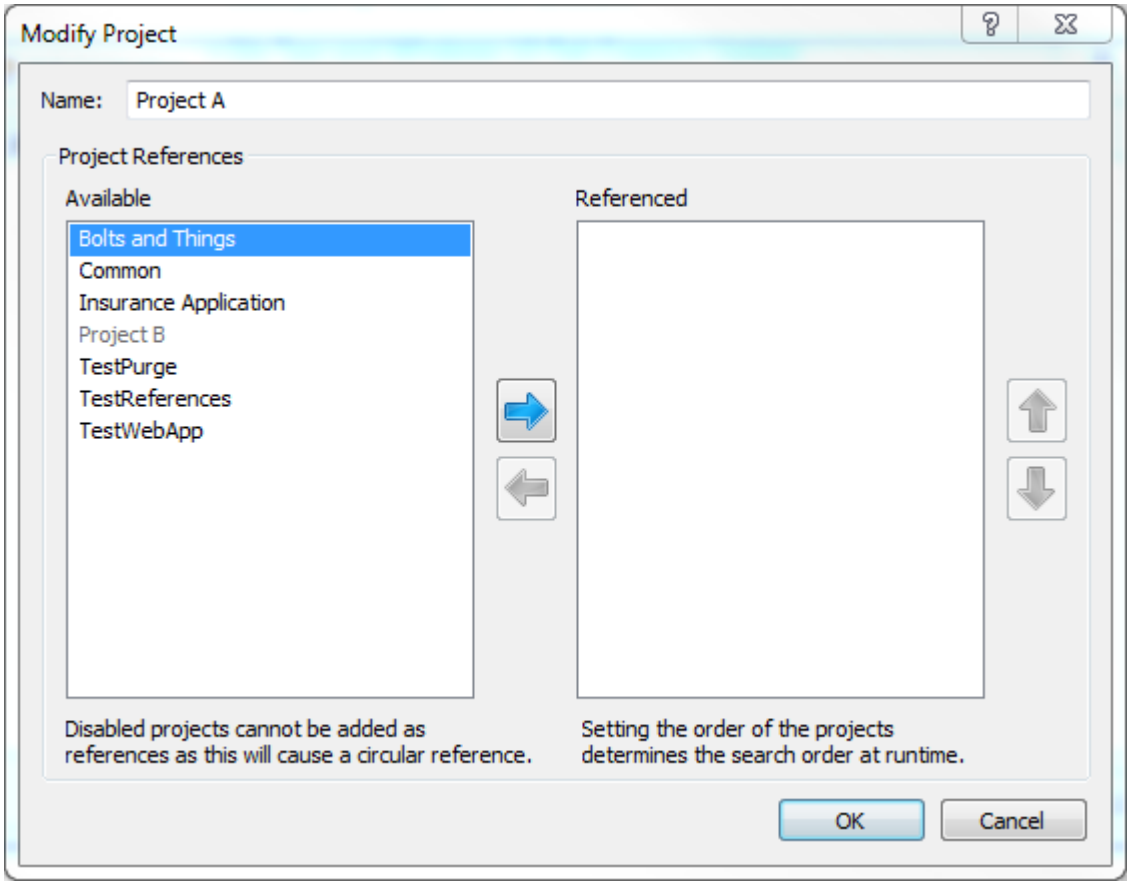
How Silk Test Workbench Avoids Circular Project References

When you add a new reference from one project to another, Silk Test Workbench needs to ensure that there are no circular references between projects. This topic describes the mechanism that Silk Test Workbench uses.

The simplest case for a circular reference is when you have two projects, Project A and Project B, where Project B references Project A.



Adding a reference from Project A to Project B would create a circular reference and therefore is not allowed by Silk Test Workbench. When you try to modify Project A, the **Modify Project** dialog shows Project B in the **Available** list as disabled.



Silk Test Workbench uses this mechanism for multiple references as well as chained references, thus successfully preventing circular project references from occurring.

Renaming a Project

You must log in as an administrator to rename projects in a Silk Test Workbench database.

1. Choose **Tools** > **Administration**.

   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.

   The **Administration** dialog box opens.
2. Click the **Projects** tab.
3. Choose the project you want to rename, and then click **Rename** to open the **Rename Project** dialog box.
4. Type the new name of the project in the **Name** text box.
5. Click **OK**.
6. If the project is referenced by other projects, specify what to do:

   - To rename the project and to update the references, click **Rename this project and update references**. All assets, to which the user has *Script Writer* access, in the projects that reference the renamed project, are scanned for references to the renamed project, and those references will be updated to use the new name of the project.
   - To rename the project without updating the references, click **Rename this project only**.

     Note: Assets in the referencing projects might not playback correctly.
   - To cancel renaming the project, click **Cancel**.

7. Click **OK** to close the **Administration** dialog The assets in any referencing projects will be updated. If you rename multiple projects without closing the **Administration** dialog, assets in referencing projects will only be updated once.

Duplicating a Project

Duplicate a project to create an exact copy of the project, for example to create a base line of the project. You must log in as an administrator to duplicate projects in a Silk Test Workbench database.

1. Choose **Tools** > **Administration**.

   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.

   The **Administration** dialog box opens.
2. Click the **Projects** tab.
3. Choose the project you want to duplicate, and then click **Duplicate**. The **Duplicate Project** dialog box appears.
4. Specify a name for the project duplicate in the **New name** field.
5. *Optional:* Check the **Copy all versions of assets** check box.

   By default, this check box is unchecked and only the active version of each asset in the original project is copied.
6. If the project references one or more other projects, select an action from the **Action** column of the referenced project to specify what to do with the project.

   - To retain the reference from the project duplicate to the referenced project, select **Retain reference**. This is the default setting.
   - To duplicate the referenced project and to reference the duplicated project from the duplicated root project select **Duplicate Project** in the **Action** column. You can change the name for the duplicated project in the **New name** column.
   - To duplicate a child referenced project, you also have to duplicate the parent project. For example, if you are duplicating project A that references project B, and project B references project C, you have to duplicate project B in order to duplicate project C.
7. Click **OK**.

   Note: This button is disabled while there are errors with referenced projects or duplicate project names.

Note: Results in the original project are not copied.

Deleting a Project

You must log in as an administrator to delete projects from the Silk Test Workbench database.

1. Choose **Tools** > **Administration**.

   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.

   The **Administration** dialog box opens.
2. Click the **Projects** tab.
3. Select the project you want to delete.
4. Click **Delete**. A message box prompts you to confirm that you want to delete the project.
5. Click **OK** to permanently delete the project.

Managing Users

Administrators must maintain the list of users authorized to access Silk Test Workbench resources and information.

Viewing which Users are Connected to the Database

To see the users that are currently connected to the Silk Test Workbench database:

In the Silk Test Workbench menu, select **Tools** > **View Connected Users**. The **Connected Users** window shows a list of all users, including columns for the user name, the full name of the user, the name of the machine from which the user is connected to the database, and the time at which the user signed into the database. The list is refreshed every five seconds.

Note: A user can appear more than once in the list, for example if the same user signs into the database from multiple machines or executes multiple copies of Silk Test Workbench on the same machine.

Adding a User

You must log in as an administrator to add users to a Silk Test Workbench database. You can override a user's default permission for a project at any time by editing their user profile.

1. Choose **Tools** > **Administration**.

   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.

   The **Administration** dialog box opens.
2. Click the **Users** tab.
3. Click **Add**. The **Add User** dialog box opens.
4. If you want to use the Windows authentication for this user, which enables Silk Test Workbench to remember the user credentials, type the domain into the **Domain** field. If you want to use the Silk Test Workbench authentication, you can leave the **Domain** field empty.
5. Enter the logon ID for the user in the **User ID** text box.
6. Enter the password for the user in the **Password** text box.
7. Enter the identical password in the **Confirm Password** text box.
8. Enter the user's name in the **Full Name** text box.
9. Use the **Type** list to specify whether the user is an administrator.
10. Use the **Default Permissions** list to assign the user a default access level for new Silk Test Workbench projects.

    Choose from the following permission types:

    - **No Access**

      User cannot display or change project information or perform recording and playback.
    - **Read Only**

      User can display project information but cannot change assets or perform recording and playback.
    - **Executor**

      User can display project information and modify results, but cannot change other assets.
    - **Script Writer**

      User can display project information and modify assets.
    - **Full Access**

User has full read/write access to all assets.

11. Click **OK**.

Deleting a User

A user must be logged in as an administrator to delete other users from the Silk Test Workbench database.

1. Choose **Tools** > **Administration**.
   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.
   The **Administration** dialog box opens.
2. Click the **Users** tab.
3. Select the user you want to delete.

   Note: You can only delete the default user *Admin* if another user with administrator rights exists.
4. Click **Delete**. A message box prompts you to confirm that you want to delete the selected user.
5. Click **OK** to permanently delete the user.

Editing a User Profile

1. Choose **Tools** > **Administration**.
   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.
   The **Administration** dialog box opens.
2. Click the **Users** tab.
3. Select the user profile to modify and click **Modify**. The **Modify User** dialog box opens.
4. Enter new information in the appropriate text boxes.
5. Click **OK** to save the changes.

Changing the Access of a User to a Project

1. Choose **Tools** > **Administration**.
   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.
   The **Administration** dialog box opens.
2. Click the **Users** tab.
3. Choose the user whose access permission you want to modify.
   The panel on the right side of the **Administration** dialog box lists the projects to which the user has access and the access level the user exercises for each project.
4. Use the **User Access** list for a project to select a new access permission level for the user.
5. Click **OK**.

Logging Users Out of the Database

As an administrator, you can turn on the asset maintenance mode and log non-administrator users out of the database to perform maintenance tasks on the assets.

1. To turn on the asset maintenance mode, select **Tools** > **Asset Maintenance Mode** in the Silk Test Workbench menu. When the asset maintenance mode is active, the following things happen:
   - The status bar in the lower part of the Silk Test Workbench window indicates that the asset maintenance mode is active.
   - Any users that are not administrators are logged out when the database maintenance mode is started. If a non-administrator user has a Silk Test Workbench dialog open, logging the user out is

not possible and the user will stay logged in to Silk Test Workbench while not being able to modify any assets in the database.
- Any open assets are saved and closed.
- Any active recording or playing back will end as soon as possible.
- While the asset maintenance mode is active, non-administrator users cannot logon to the Silk Test Workbench database.

2. To log out as an administrator from Silk Test Workbench while the database is in asset maintenance mode, you have specify whether to deactivate the asset maintenance mode before logging out. Click **Yes** to deactivate the asset maintenance mode.

Managing Groups

Enables groups of Windows domain accounts to login to the database with the same permissions.

If a group has a shared database, you can simply add the group to enable all group members to login to the database with the same permissions. If a new member is added to the group, it automatically gains access to the database, and if a member is removed from the group, it no longer has access to the database.

To enable all users on the local machine to access the database, add **BUILTIN\All Users** to the groups list. For new Access databases, this group is added to the groups list by default, to provide an easier start for new users.

Adding a Group

Add a group profile to enable a group that shares a Silk Test Workbench database to access the database by using the same credentials.

1. Choose **Tools** > **Administration**.

   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.

   The **Administration** dialog box opens.
2. Click the **Groups** tab.
3. Click **Add**. The **Add Group** dialog box opens.
4. If you want to use the Windows authentication for this group, which enables Silk Test Workbench to remember the group credentials, type the domain into the **Domain** field. If you want to use the Silk Test Workbench authentication, you can leave the **Domain** field empty.
5. Use the **Type** list to specify whether the group is a group of administrators.
6. Use the **Default Permissions** list to assign the group a default access level for new Silk Test Workbench projects.

   Choose from the following permission types:

   - **No Access**

     User cannot display or change project information or perform recording and playback.
   - **Read Only**

     User can display project information but cannot change assets or perform recording and playback.
   - **Executor**

     User can display project information and modify results, but cannot change other assets.
   - **Script Writer**

     User can display project information and modify assets.
   - **Full Access**

     User has full read/write access to all assets.

7. Click **OK**.

Deleting a Group

A user must be logged in as an administrator to delete groups from the Silk Test Workbench database.

1. Choose **Tools** > **Administration**.
   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.
   The **Administration** dialog box opens.
2. Click the **Groups** tab.
3. Select the group that you want to delete.
4. Click **Delete**. A message box prompts you to confirm that you want to delete the selected group.
5. Click **OK** to permanently delete the group.

Modifying a Group

1. Choose **Tools** > **Administration**.
   Or, you can click **Administration** in the **Tasks** pane of the **Start Screen**.
   The **Administration** dialog box opens.
2. Click the **Groups** tab.
3. Select the group that you want to modify and click **Modify**. The **Modify Group** dialog box opens.
4. Enter new information in the appropriate text boxes.
5. Select the user type and permissions from the appropriate list boxes.
6. Click **OK** to save the changes.

UI Topics

Silk Test Workbench's desktop is the starting point for all test activities. With the exception of the menu bar, you can control the display and appearance of the windows, panes, and other features on the desktop.

Start Screen

The **Start Screen** is your launching point into the functionality of Silk Test Workbench, enabling you to quickly begin creating test solutions for all your applications.

To enable you to quickly start testing your applications with minimal effort, the **Start Screen** includes the following areas:

Area	Description
Start	Enables you to record or open visual tests, .NET scripts, and keyword-driven tests with a single click.
Assets	Enables you to open recently used visual tests, .NET scripts, and keyword-driven tests with a single click. The recent assets list is updated in real time so any changes or updates will move the current asset to the top of the list. You can play back an asset or view eventual results of previous executions of the asset by clicking on the corresponding icons to the right of the asset.  **Note:** The **Start Screen** settings under **Tools** > **Options** > **Start Screen** allow you to control how many items display in the recent assets list.
What's New	Lists new features and enhancements that are included in the current version of Silk Test Workbench. Also lists available updates for Silk Test Workbench.  **Note:** If you have opted not to display the start screen when you start Silk Test Workbench, you can check for available updates by clicking **Help** > **Check for Product Update**.
Get Started	Provides quick links to commonly used tasks and useful help topics to get you quickly productive.
Resources	Provides links to information that is related to Silk Test Workbench, for example to the support knowledge base, the Silk Test community, and so on.
News & Community	Provides links to the latest forum entries and blog posts on the Silk Test community.
Flags	Displays the test steps in visual tests and results that are flagged and assigned for follow-up. Use the **Flags** pane to collaborate with other testers and other users of the test projects in the database, by exchanging information about test suites and test projects. By default, flags are grouped by the date they were created and the asset they appear in. To quickly view information about a flag, move your pointer over a flag. A ToolTip displays containing the flag description, modification data, the project, and the ID to whom the flag is assigned. You can change how flags are displayed by modifying the **Start Screen Flag** options under **Tools** > **Options** > **Start Screen** > **Flags**. Double-click a flag in the **Flags** pane to open its associated visual test or result.

Area	Description

Note: The **Flags** pane is only visible in the **Start Screen** if at least one flag is assigned to you or if at least one flag is filtered by the settings to be shown for you.

Click **Close Window** in the upper-right corner of the **Start Screen** to close the **Start Screen**.

Tip: You can always access the **Start Screen** from the main screen by clicking **Start Screen** in the toolbar. You can also press `Ctrl+Alt+S`, or choose **View** > **Start Screen** at any time to display it.

Note: The user interface works with standard Windows small and large font sizes. Using a custom font size may result in the inaccurate display of text in the user interface.

Silk Test Workbench Menus

This section describes the Silk Test Workbench menus.

Menu commands may be enabled or disabled depending on the type of window, asset, or testing mode that is currently in use.

File Menu

The **File** menu contains the following commands:

Command (Shortcut)	Description
New (Ctrl+N)	Displays the **New** window, where you can create any type of asset.
Open (Ctrl+O)	Opens the **Asset Browser** to allow you to open an asset. Highlights the asset type last worked with.
Close	Closes the window in the desktop that is currently in focus.
Save as new version (Ctrl +S)	• Saves the current state of the asset as a new version of the asset and assigns an incremental numerical identifier to the version. The shortcut key combination is enabled if the default save behavior is set to **Save as new version**. For more information, see *Setting the Default Behavior for Saving Assets*. • Saves the current state of an asset as the current version of the asset. The shortcut key combination is enabled if the default save behavior is set to **Save as current version**. For more information, see *Setting the Default Behavior for Saving Assets*. This command is disabled the first time you save an asset.
Save as current version (Ctrl+S)	Saves the current state of an asset as the current version of the asset. The shortcut key combination is enabled if the default save behavior is set to **Save as current version**. For more information, see *Setting the Default Behavior for Saving Assets*. This command is disabled the first time you save an asset.
Save As	Opens the **Save As** dialog box to store the current asset under a different name.
Save All (Ctrl+Shift+L)	• Saves the current states of all open assets as new versions of the assets and assigns an incremental numerical identifier to the version for each asset. The shortcut key combination is enabled if the default save behavior is set to **Save as new version**. For more information, see *Setting the Default Behavior for Saving Assets*. • Saves the current states of all open assets as the current version of the assets. The shortcut key combination is enabled if the default save behavior is set to **Save as**

Command (Shortcut)	Description
	current version. For more information, see *Setting the Default Behavior for Saving Assets*. This command is disabled the first time you save an asset.
Duplicate	Select an asset in the **Asset Browser**, then use this menu selection to duplicate the asset to a new name.
Rename	Select an asset in the **Asset Browser**, then use this menu selection to rename the selected asset.
Delete (Del)	Deletes the selected item.
Show All Versions	Select an asset in the **Asset Browser**, then use this menu selection to view all versions of the asset.
Summary Information (Alt+Enter)	Display the **Summary Information** dialog box from within a visual test or script.
Page Setup	Displays the **Page Setup** provided by the browser control. Settings in this dialog box are applied to the currently selected asset and are visible in the print preview dialog.
Print Preview	Displays a **Print Preview** dialog box, where you can see the layout of the current asset before printing it.
Print (Ctrl+P)	Displays the print dialog box, where you can print the selected asset.
Export Assets	Export assets from one database to another by using the **Asset Export** wizard. For additional information, see *Exporting Assets*.
Import Assets	Import assets from one database to another by using the **Asset Import** wizard. For additional information, see *Importing Assets*.
Recent Assets	Displays a submenu of the most recently accessed test assets.
Exit	Exits Silk Test Workbench.

Edit Menu

The **Edit** menu contains the following commands:

Command	Description
Undo (Ctrl+Z)	Reverses the previous edit in a test script. Not available for visual tests.
Redo (Ctrl+Y)	Reverses the previous **Undo** action in a test script. Not available for visual tests.
Cut (Ctrl+X)	Cuts a selected item and moves it to the clipboard.
Copy (Ctrl+C)	Copies a selected item.
Paste (Ctrl+V)	Pastes a selected item.
Select All (Ctrl+A)	Selects all test steps in the current visual test or code lines in the current test script.
Find and Replace	Find specific code within a script and replace it if necessary.
Go To Line (Ctrl+G)	Goes to a specific line in a script.
Bookmarks	Toggle bookmarks on or off, navigate to the previous or next bookmark, or clear bookmarks in scripts.

Command	Description
Comment Line (Ctrl+K)	Disable one or more currently selected lines in a script by commenting them out. You can use this shortcut to comment out entire blocks of code. You can also use **Ctrl+F11**.
Uncomment Line (Ctrl+Shift+K)	Re-enable one or more selected lines in a script by un-commenting them.
Flag	Edits or removes the currently selected flag.
Update Screen	Updates screen information from the test application for a selected test step. The screen from the test application must be running, and the step requiring update must be selected.
Criteria	Displays the **Criteria** dialog box, which lets you set the percentage of verifications that must pass in a playback for the overall verifications to pass.
Manage Filters	Displays the **Manage Filters** dialog box, which is used to filter the data displayed for results. Filters are applied to both the **Summary** view and **Details** view regardless of what view is currently visible.

Actions Menu

The **Actions** menu contains the following commands:

Command	Description
Record	Records all user actions into the active visual test or test script, at the current location.
Pause	Suspends a visual test or test script that is currently being played back and highlights the point where the execution was halted.
Playback (F5)	Playback the active visual test or script.
Stop	Aborts the playback of the currently active visual test or script.
Execute Base State	Returns the application under test to the base state, which is a known, stable state in which the application should be before a test begins execution. For additional information, see *Base State*.
Show Differences	Compares the differences between two selected assets.
Update Screen	Refreshes screen information from the test application for a selected test step. The screen from the test application must be running, and the step requiring update must be selected.

View Menu

The **View** menu contains the following commands:

Command	Description
Go To	Contains a submenu with commands to navigate to one or more of the following: **Details page** – Navigates from a selected step in the **Passed** or **Failed** tab of a result to the same selected step in the **Details** tab of a result. **Visual test Window** – Navigates from the result window to the visual test window.

Command	Description
	Result Window – Navigates from the visual test window to the result window.
Logic Toolbox	Displays the Test Logic Toolbox, which provides links to the **Test Logic Designer**.
Open in New Window	Displays a selected image in the **Screen Preview** or a selected result in a new window.
Test Steps	Toggles display of specific steps in the **Test Steps** or **Results** window of the **Visual Navigator**.
	Steps Only – Displays steps that perform test actions against the test application.
	Screens Only – Displays steps that show screens associated with the test.
	Steps and Screens – Displays steps that both perform actions and show screens.
	Step Description – Displays any descriptions entered against the displayed test steps.
Columns	Displays result information in the **Details** tab of the **Results** window.
	Basic View – Displays the standard **Test Step** pane information with the additional columns of **Result** and **Result Detail**.
	Advanced View – Displays detailed information for each step.
Start Screen (Ctrl+Alt+S)	Displays the **Start Screen**.
Asset Browser (Ctrl+Alt+A)	Opens the **Asset Browser**.
Error List	Displays errors in red in the **Test Steps** pane.
Output	Displays the **Output** window.
Includes	Displays the **Includes** window.
Keywords	Displays the **Keywords Editor**, which lists all keywords in the active project as well as any keywords that have been defined in an integrated Silk Central instance.
Refresh	Refreshes the window.
Zoom	Magnifies or reduces the size of the current image in the **Screen Preview**. Enabled when the application window is in focus.
Toolbars	Displays or hides the toolbars.
Status Bar	Displays the status bar, which shows current session information.
Restore Layout	Restores the layout in the **Visual Navigator** to the original view setting.

Insert Menu

The **Insert** menu contains the following commands:

Command	Description
Test Logic	Creates logic for the current visual test. For additional information, see *Toolbars* or *Test Logic*.
Control From	Inserts a control into a visual test from one of the following:

Command	Description
	• The application under test. • The **Screen Preview**. • The **Identify Object** dialog.
File	Inserts an external file into a script. Enabled when a script is active.
Visual test	Opens the **Browse for Visual Test** dialog box to add a visual test within the active visual test. Enabled when a visual test is active.
.NET Script	Opens the **Browse for .NET Script** dialog box to add a .NET script within the active visual test. Enabled when a visual test is active.
External Program	Starts an application or program at a selected point during the active visual test playback.
Expression	Displays the **Expression Designer**, which lets you insert a formula into the active visual test.
Verification asset	Inserts an existing verification asset into a visual test at the current step. Currently only image verifications are supported as verification assets. For additional information, see *Image Verifications*.
Result Comment	Inserts a result comment at the current step of a visual test. For additional information, see *Inserting a Result Comment in a Visual Test*.
Label	Inserts a blank, editable step into a visual test with no property values.
Error Handling	Inserts error handling into a visual test at the current step.
Message Box	Inserts a message box test step.
Comment	Inserts a comment into the visual test at the current step.
Playback Setting	Sets or retrieves playback settings and their values into the current test.
Property from Control	Inserts an empty step that can be used to capture a property from control in the test application and set the property to a variable. In the **Properties** pane of the step, use the **Identify** buttons of the **Locator** field to capture the control and select a specific property of the control.
Program Flow	Inserts steps that control test flow or makes tests more reliable.
Synchronization and Timing	Inserts a test step to wait for an object, delay visual test playback for a specified time, or create a timer for playback.
SAP eCATT	• **Get Argument** • **Set Argument** Allows passing variables between a visual test and an SAP eCATT asset. Enabled when SAP is installed on the same computer as Silk Test Workbench. For additional information, see *Working with SAP eCATT*.
ActiveData	Provides the following actions: **New** — Creates a new ActiveData asset and associates it with the visual test. For additional information, see *Associating an ActiveData Asset with a Visual Test*. **Associate Existing** — Associates an existing ActiveData asset with the visual test. For additional information, see *Associating an ActiveData Asset with a Visual Test*.

Command		Description
	Save Now	Inserts a step into the visual test to immediately save an ActiveData file. For additional information, see *Immediately Saving Updated ActiveData in a Visual Test*.
	Cancel Save	Cancels a previous request to save an ActiveData file.
Variable		Adds local variables, or sets or retrieves global parameters for a visual test. For additional information, see *Variables in Visual Tests*.
Flag		Sets a flag or note against a selected test step. For additional information, see *Flags*.
Update Screen		Refreshes screen information from the test application for a selected test step. The screen from the test application must be running, and the step requiring update must be selected.

Debug Menu

For information on debugging tests, see *Debugging Tests*. The **Debug** menu contains the following commands:

Command (Shortcut)	Availability	Description
Compile (**F7**)	• Scripts	Checks that the script syntax meets all preconditions for compiling, compiles the code, and shows any errors that occur.
Step Into (**F8**)	• Scripts • Visual tests	Plays back a visual step one test step at a time or a script one statement at a time.
Step Over (**Shift+F8**)	• Scripts • Visual tests	Plays back an embedded visual test or script in its entirety and suspends in debug mode at the next step in the original visual test or script. Also steps over a function call when pressed while in a VB .NET script.
Step Out (**Ctrl+Shift+F8**)	• Scripts • Visual tests	Executes all remaining steps in a visual test being played back from another visual test, then suspends playback at the next step in the original visual test. For scripts, executes all remaining code in a procedure as if it were a single statement, and exits to the next statement in the procedure that caused the procedure to be initially called.
Run To Cursor (**Ctrl+F8**)	• Scripts • Visual tests	Plays back a visual test to the currently selected test step or a script to the currently selected statement.
Run From Cursor	• Visual tests	Plays back the visual test from the currently selected test step.
Set/Clear Breakpoint (**F9**)	• Scripts • Visual tests	Sets a breakpoint at the selected step or line of code or clears an existing breakpoint from the selected step or line of code.
Clear All Breakpoints (**Ctrl+Shift+F9**)	• Scripts • Visual tests	Clears all breakpoints in all open visual tests and VB .NET scripts.
Set Run pointer/next Statement (**Ctrl+F9**)	• Scripts • Visual tests	Executes playback to the next statement or pointer. Enabled when in debug mode.

Command (Shortcut)	Availability	Description
Local Variables	• Scripts • Visual tests	Displays a window that lists all variables and their values for the current visual test or script and any embedded visual tests or scripts.

Tools Menu

The **Tools** menu contains the following commands:

Command	Description
Change Database	Displays the **Logon** dialog box to log onto another database.
Change Password	Displays the **Change Password** dialog box to change the logon password for the current user.
Identify Object (Ctrl+Shift+I)	Opens the **Identify Object** dialog box where you can record a locator for a selected object.
Administration	Manages users and access to projects.
Options	Displays configuration settings and lets you customize those settings.
Upload Keyword Library	Uploads a keyword library to Silk Central. Enabled only when Silk Test Workbench is integrated with Silk Central. For additional information, see *Uploading a Keyword Library to Silk Central*.
Manage Custom Controls	Displays the **Manage Custom Controls** dialog box.
Edit Application Configurations	Displays the **Edit Application Configurations** dialog box, which enables you to specify the application that is tested by keyword-driven tests in a specific project and to configure how this application is started. For additional information, see *Editing Application Configurations*.
Edit Remote Locations	Displays the **Remote Locations** dialog box which enables you to specify any remote locations on which you want to access browsers and devices. For additional information, see *Editing Remote Locations*.
Customize Tools	Adds menu items to the **Tools** menu.
SAP eCATT	Allows access into the commands to work with SAP eCATT. Displayed only when Silk Test Workbench is integrated with SAP eCATT. For additional information, see *Working with SAP eCATT*.

Customizing the Tools Menu

Silk Test Workbench lets you add menu items to its **Tools** menu. These menu items make it easy to open a program that you use often. Adding menu items is similar to using shortcuts, as they offer are quick ways to get to the programs and documents that you use often. For example, if you create a custom menu item for Notepad, you can start it from the **Tools** menu without having to stop working in Silk Test Workbench.

Do not click **Add** after you enter the information for the new command. This adds another blank entry as **(new tool)** which will then be visible as an item in the **Tools** menu.

1. Click **Tools** > **Customize Tools**. The **Customize** dialog box opens.

2. To create a new item, click **Add**. The **Menu Contents** list displays a blank entry as **(new tool)**. The **(new tool)** entry also displays in the **Menu Text** text box.
3. In the **Menu Text** text box, type a name for the menu item.

   This name will appear in the **Tools** menu. For example, if you want to the menu item to start Notepad, you can type `Notepad`.
4. In the **Command** text box, click **Browse** and navigate to and select the file that executes the menu item.

   To select the file, highlight it and click **Open** in the **Select Program** dialog box.

   You can also type the path and file name in the **Command** text box if you know it.
5. In the **Arguments** text box, type any parameters needed to execute the file in the **Command** text box.

   Leave this text box empty if there are no command requirements.
6. In the **Initial Directory** text box, enter the path and directory that contains the original item or any related files needed by the file executed by the menu item.

   Sometimes, programs need to use files from other locations. You may need to specify the folder where these files are located so that the file executed by the menu item can find them.
7. Click **OK**. The **Customize** dialog box closes, and the item is added to the **Tools** menu.

Window Menu

The **Window** menu contains the following commands:

Command	Description
Tile Horizontally	Displays all open windows horizontally.
Tile Vertically	Displays all open windows vertically.
Cascade	Displays all open windows layered in a cascading pattern.
Arrange Icons	Arranges minimized scripts, visual tests, and so on along the bottom of the window.
List of open files	Displays a submenu of the open visual tests, results, scripts, or windows, such as the **Asset Browser** or **Start Screen**.

Help Menu

The **Help** menu contains the following commands:

Command	Description
Contents	Opens help to its table of contents.
Search	Opens the help **Search** tab.
Index	Opens the **Index** tab in the help.
Tutorial	Opens the Silk Test Workbench tutorial.
Check for Product Update	Enables you to check if a newer version of Silk Test Workbench is available.
About Silk Test Workbench	Displays Silk Test Workbench product information.

Toolbars

Click **View** > **Toolbars** in the menu and check or uncheck the check box to the left of the toolbar to display or hide the toolbar. The following toolbars are available in Silk Test Workbench:

File Provides the following actions for asset management:

Button	Description
New (Ctrl+N)	Displays the **New** window, where you can create any type of asset.
Open (Ctrl+O)	Opens the **Asset Browser** to allow you to open an asset. Highlights the asset type last worked with.
Save as new version (Ctrl+S)	• Saves the current state of the asset as a new version of the asset and assigns an incremental numerical identifier to the version. The shortcut key combination is enabled if the default save behavior is set to **Save as new version**. For more information, see *Setting the Default Behavior for Saving Assets*. • Saves the current state of an asset as the current version of the asset. The shortcut key combination is enabled if the default save behavior is set to **Save as current version**. For more information, see *Setting the Default Behavior for Saving Assets*. This command is disabled the first time you save an asset.
Save All (Ctrl+Shift+L)	• Saves the current states of all open assets as new versions of the assets and assigns an incremental numerical identifier to the version for each asset. The shortcut key combination is enabled if the default save behavior is set to **Save as new version**. For more information, see *Setting the Default Behavior for Saving Assets*. • Saves the current states of all open assets as the current version of the assets. The shortcut key combination is enabled if the default save behavior is set to **Save as current version**. For more information, see *Setting the Default Behavior for Saving Assets*. This command is disabled the first time you save an asset.
Print (Ctrl+P)	Displays the print dialog box, where you can print the selected asset.
Print Preview	Displays a **Print Preview** dialog box, where you can see the layout of the current asset before printing it.
Index	Opens the **Index** tab in the help.

Edit Provides the following actions for editing:

Button	Description
Cut (Ctrl+X)	Cuts a selected item and moves it to the clipboard.

Button	Description
Copy (Ctrl+C)	Copies a selected item.
Paste (Ctrl+V)	Pastes a selected item.
Undo (Ctrl+Z)	Reverses the previous edit in a test script. Not available for visual tests.
Redo (Ctrl+Y)	Reverses the previous **Undo** action in a test script. Not available for visual tests.
Comment Line (Ctrl+K)	Disable one or more currently selected lines in a script by commenting them out. You can use this shortcut to comment out entire blocks of code. You can also use **Ctrl+F11**.
Uncomment Line (Ctrl+Shift+K)	Re-enable one or more selected lines in a script by un-commenting them.

Actions Provides the following actions for the recording process:

Button	Description
Record	Records all user actions into the active visual test or test script, at the current location.
Playback (F5)	Playback the active visual test or script.
Pause	Suspends a visual test or test script that is currently being played back and highlights the point where the execution was halted.
Stop	Aborts the playback of the currently active visual test or script.
Identify Object (Ctrl+Shift+I)	Opens the **Identify Object** dialog box where you can record a locator for a selected object.

View Displays the **Asset Browser** and the **Start Screen**.

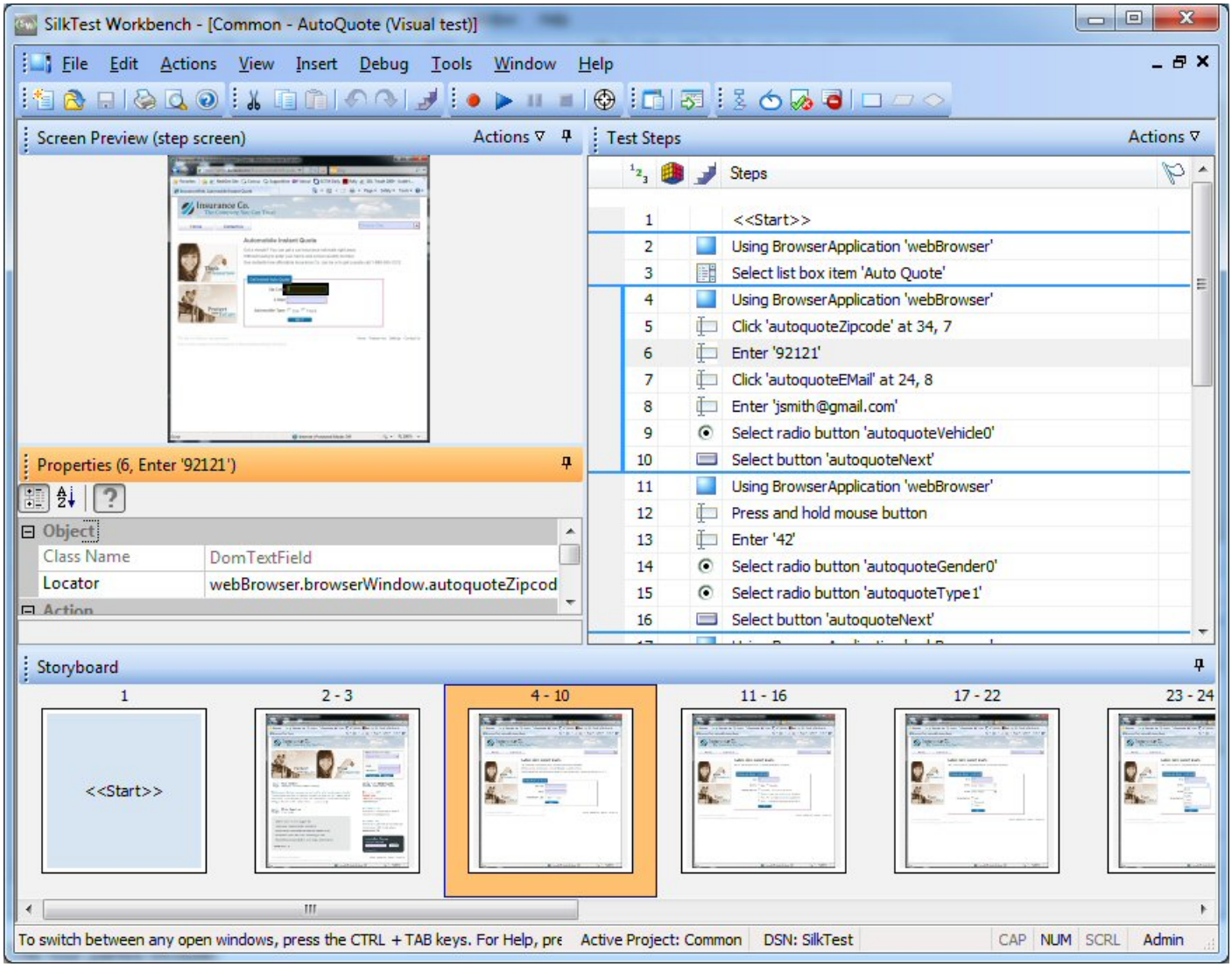
Test Logic Provides the following actions for adding logic to visual tests:

Button	Description
Create Decision Type Logic	Decision logic evaluates a user-defined condition, and then runs a sequence of test steps depending on the return value of the condition.
Create Repetition Type Logic	Repetition logic repeats a sequence of test steps a set number of times while or until a user-defined condition is true.
Create Verification Type Logic (Ctrl+Alt)	Verification logic evaluates a user-defined condition, and then sends a pass/fail message to the result of a visual test or script.
Create Playback Error Handler Logic	Error handling logic eliminates the need for debugging playback errors by telling the visual test to detect errors, and how to respond to them when they occur.

Button	Description
Insert An If Type Logic Item	Add conditional logic to the active visual test by enclosing the selected test steps between an `If` and an `End If` test step. The test steps are only executed if the condition is met.
Insert An Else If Type Logic Item	Can only be used between an `If` or `Else If` test step and an `End If` test step in a visual test. The test steps between `Else If` and `End If` are executed if the conditions of any preceding `If` or `Else If` test steps are not met while the condition of the `Else If` test step is met.
Insert An Else Type Logic Item	Can only be used between an `If` or `Else If` test step and an `End If` test step in a visual test. The test steps between `Else` and `End If` are executed if the conditions of any preceding `If` or `Else If` test steps are not met.

Visual Navigator

The **Visual Navigator** graphically represents the elements of a visual test and allows you to interact with each element through a point-and-click interface. When viewed in the **Visual Navigator**, a visual test is represented by information displayed in four panes that collectively provide a comprehensive view of each step in a visual test.



The four panes include:

Test Steps	Lists each step of a visual test in clear non-technical language.
Screen Preview	Displays a snapshot of the application under test as it appears when a step executes during playback of a visual test.
Properties	Displays the properties of a step in a visual test.
Storyboard	Displays the flow of a visual test through the use of thumbnail images, which represent the logical groups of steps in a visual test.

Note:

Silk Test Workbench takes snapshots under the following circumstances:

- Before every automation test step during recording.

  Note: For SAP applications, snapshots occur when the screen changes rather than before every automation test step.
- When executing a `Using` step in a visual test, a snapshot of the result.
- When a playback error occurs.

The **Screen Preview**, **Storyboard**, and **Properties** panes are synchronized with the **Test Steps** pane and display information specific to a selected step in the **Test Steps** pane. In this way, you can easily view all aspects of a step by selecting a step in the **Test Steps** pane, and then viewing information about the step in the other panes.

In addition to viewing a visual test, the **Visual Navigator** also allows you to enhance or update an existing visual test by using the **Screen Preview** and **Properties** panes. For example, in the **Properties** pane, after recording a visual test, you can change the literal value of a recorded property by replacing it with a variable. Additionally, to quickly update a visual test when changes occur in the application under test, you can update previously captured screens using the **Update Screen** feature of the **Screen Preview**.

The **Visual Navigator** also displays the playback result of a visual test using the same panes as those used for a visual test. For a result, the panes have additional functionality and appear in the **Result** window, which contains toolbar options and several tabs that display different views of result content. Examples of additional functionality specific to a result include the ability to see the pass or fail status of each step in the **Test Steps** pane. Additionally, in the **Screen Preview**, you can see a comparison of the differences between the screens captured during recording and screens captured during playback, and then update the existing visual test without accessing the test application.

Visual Navigator: Test Steps Pane

The **Test Steps** pane is the central area used for viewing and maintaining visual tests and the results of visual tests and scripts. The **Test Steps** pane displays a line-by-line representation of the recorded or inserted actions that make up a visual test. Each action is represented in a separate line called a step. Steps describe the actions taken to test a test application in clear non-technical language. Each step corresponds to an action performed against the test application or actions used to manage the test.

Steps that interact with a test application are represented by images of the corresponding application screen elements in the **Screen Preview**. Each step is grouped to also correspond to a thumbnail in the **Storyboard**.

Each step contains properties that determine specifics for the action of the step. A step's properties display in the **Properties** pane.

From the **Test Steps** pane of a visual test, you can review a recorded visual test and add actions to improve test reliability, flexibility, validation, function and flow, including:

- Add test logic
- Specify how the visual test is played back
- Execute other visual tests and scripts
- Debug visual tests
- Provide error handling

You can customize the **Test Steps** pane to display only the steps within a visual test that you want to view.

The **Test Steps** pane appearing in the **Result** window offers additional information pertaining specifically to test results. With the exception of the **Summary** tab (which does not display the **Test Steps** pane), this information is displayed in a **Basic View** and an **Advanced View**.

The standard view of the **Test Steps** pane contains the following columns:

Column	Description
(Number)*	Represents the sequential order in which steps are recorded and played back. Steps display in this sequential order by default. Any steps that have breakpoints added for debugging display a breakpoint in the column next to this column.

Column	Description
	If you change the view from the default view (Steps and Screens) to the Screens only or the Steps only view, the numbering scheme sequentially skips the steps or screens to indicate the gap in the recording sequence.
 (Logic)*	Displays icons representing the type of logic for the step, if the step contains logic.
 (Step Type)*	Displays icons representing the type of action being executed in the step.
Steps	Describes the action being taken for the step.
 (Flag)*	Displays the **Assigned Flag** icon. The **Assigned Flag** icon indicates that the associated step is flagged to appear in the **Test Steps** pane and optionally on the **Start Screen** of the assigned user. Additionally, you can move your pointer over the icon to display a ToolTip containing the flag description, modified date, and assigned date.
 (Step Description)*	Displays a user-defined step description. This column does not display in the default view, but can be shown by clicking **Actions** from the **Test Steps** pane title bar and then **View** > **Step Description**. To create a step description, select a step. In the **Properties** pane, update the **Step description** property. A **Step Description** icon appears in this column indicating that the step has a description. Either move your pointer over the icon to display a ToolTip containing the description or select the step and read the description in the **Properties** pane.

* This column is not resizable.

Basic View of the Result Window

Column	Description
Result	Displays the execution status of a step in a visual test or code line in a script.
Result Detail	Displays the details of the execution status of a step in a visual test or code line in a script.

Advanced View of the Result Window

Column	Description
Line Number	Provides the line number of the code line in a script or step number in a visual test. Use in conjunction with the Command Details to locate the specific code line or step number.
Command Details	Displays the actual code line text. Use in conjunction with the Line Number to locate the specific code line in the script or step number in a visual test.
Date/Time	Displays the time-stamp of when the test step or code line was run.
Milliseconds	Displays the duration of the test step or code line during the run.
Name	Displays the name of the visual test or script.
User Name	Lists the logon identifier of the person who ran the test.
Machine Name	Lists the unique identifier of the device used to run the test.

Test Step Icons

Icons graphically represent commonly performed actions in test steps in a visual test. Graphics display in the **Test Step** column of the **Test Steps** window. Commonly used actions and their icons are shown in the following table.

Step Type	Icon	Description
Screen capture		Indicates that a screen capture occurs for this step during playback.
Object classes	various	Each UI object class has a unique test step icon. For instance, the `Window` class icon looks like  while the `TextField` class looks like  .
Wait for object		Indicates that a `Wait for object` property has been inserted.
Decision logic		Executes an If statement, Else If, or End If statement.
Repeat test logic		Executes a sequence of steps a specified number of times.
Verification logic		Validates the condition stated in the `Condition` property for the step.
Error handling logic		Performs error handling.
Visual test		Plays back the named visual test.
External program		Launches the named external program according to the step's attributes.
Expression		Contains an expression or formula.
Result comment		Contains a comment for a result.
Label		Contains a label.

Test Step Types

Each step in a visual test performs a specific action, whether executing an action against a control in a test application, evaluating complex conditions, or simply book-marking a place in the visual test. The **Steps** column of the **Test Steps** pane describes the action that the corresponding step performs.

General test steps are test steps that do not perform an action against a control in a test application. The general test step types that are available for visual tests are shown in the following table:

Beginning Step Text	Description
<<Start>> and <<End>>	The first and last steps of any visual test. The properties of any passed in variable to be used in the visual test are displayed in the `<<Start>>`step properties.
Verify	Validates the condition stated in the `Condition` property for the step.
If / Else / Else If / End If	Performs decision logic that evaluates a user-defined condition, and then runs a sequence of test steps depending on the return value of the condition. When an `If` test step is inserted into a visual test, the corresponding `End If` test step is also inserted.
Playback visual test	Plays back the named visual test using the Playback settings property associated with the step.
Playback .NET Script	Plays back the named script using the Playback settings property associated with the step.
Run program	Starts an external program based on `Command` property settings for the step.

Beginning Step Text	Description
Expression	Executes and evaluates an expression in a formula and sets its result to a variable.
Result Comment	Creates the specified comment and sends it to the result when the visual test is played back. Comments can be either text, the value of an expression, an ActiveData file, or the result of an expression, and display in the **Command Details** column of the test **Results Details** tab.
Label [label name]	Sets a step as a label. The text of the step is the name of the label as set in the step's `Label` property. Use labels to bookmark specific areas or steps in a test to control test flow and playback.
On Error	Executes error handling. The `Action to Take` property for the step determines how the test behaves if an error occurs while the error handling is in effect. The `Asset` property for the step can be used to playback another test if an error occurs while the error handling is in effect.
Message box	Displays a message box to the tester during playback. Message box content is determined by the property values set for the step.
Set Playback Setting	Changes a playback setting from the run environment to a specific value. The value is set in the `Playback Setting` property for the test step. The value is only in effect for the visual test playback or until it is reset.
Get Playback Setting	Retrieves a playback setting from the run environment and allows the setting to be set to a variable.
Get the "" property of the control	Inserts an empty step that can be used to capture a property from control in the test application and set the property to a variable. In the **Properties** pane, use the **Identify** buttons to capture the control and select a specific property of the control.
Go to label	Jumps to the specified label step for playback.
Wait for	Monitors for the existence or disappearance of a control. The control to monitor can be represented as a literal, variable, expression, or ActiveData. The `Timeout` property determines the wait duration.
Stop	Completely halts playback. Stop is normally used in conditional logic, such as "If" steps.
Delay for	Pauses playback for the duration specified by the `Delay Amount` property for the step. The `Delay Type` property for the step determines whether the duration amount is seconds or milliseconds.
Start timer number	Creates a timer and starts it. Use timers to learn how long it takes to playback all or part of a visual test. Multiple timers can be used throughout a visual test.
Stop timer number	Stops the timer specified by the `Timer Number` property for the step. Use timers to learn how long it takes to playback all or part of a visual test. Multiple timers can be used throughout a visual test.
Resume Timer	Resumes the stopped timer named by the `Timer Number` property for the step.
Get eCATT argument	Retrieves an argument value from an SAP eCATT argument container. The value is passed from an eCATT script to the visual test. The step connects a variable to a predefined import argument that is passed from a SAP eCATT script.

Beginning Step Text	Description
	An eCATT argument value can be retrieved when SAP eCATT is installed on a computer that has Silk Test Workbench installed.
Set eCATT argument	Sets an argument value to pass to an SAP eCATT script. The step connects a variable to a predefined export argument that can be passed to an SAP eCATT script. An eCATT argument value can be set when SAP eCATT is installed on a computer that has Silk Test Workbench installed.
Save values immediately to ActiveData file	Saves any associated data that has been updated for the specified ActiveData asset when the step executes.
Do not save values to ActiveData file	Prevents any data that has been updated from the specified ActiveData file to be saved back to the file up to the point the step executes. If a Save values immediately to ActiveData file step is executed after this step for the same ActiveData file, and data for the file that has been updated since this step executes will be saved back to the file.
Set global variable	Defines a global variable by naming it and initializing its value.
Get contents of global variable	Retrieves the value of a global variable and sets it to a local variable for use in a visual test.

Visual Navigator: Screen Preview

The **Screen Preview** displays a captured image of the test application for each step in the **Test Steps** pane that interacts with a control. The captured image can show the full desktop, the application window, or only the active window. Captured images represent the state of the application before its associated step is executed.

The **Screen Preview** synchronizes with the **Test Steps** pane. When you select a step that interacts with a control, the **Screen Preview** displays an image of the test application screen and highlights the control being automated in the selected step. If the highlighted step interacts with a control in the image, the control highlights with a rectangle around the control. Also, clicking a control in a captured image highlights the first sequential step that interacts with that control if any exist.

When working with a visual test in the **Visual Navigator**, you can interact with images in the **Screen Preview** to create steps without requiring access to the test application.

When the **Test Steps** pane in the **Visual Navigator** is empty, the **Screen Preview** displays a watermark image that has instructions on how to start recording.

Viewing Captured Test Application Screens

To view captured images of the test application in the **Screen Preview**, select the step in the **Test Steps** pane that includes the captured image you want to view. Once the captured image appears, you can view the image at a different size or view the image in a separate window.

1. Select the step that corresponds to the captured image that you want to view at a different size.
2. From the **Screen Preview**, click **Actions**, and then one of the following commands:
   - **Zoom** – Displays the captured image in the existing **Screen Preview** pane. Select a percentage (10% to 300% of its captured size) or click **Fit to Window** to size the image so it fits within the current size of the **Screen Preview**.
   - **Open in New Window** – Displays the captured screen in a separate, resizable window from the **Screen Preview** pane.

Visual Navigator: Properties

The **Properties** pane of the **Visual Navigator** displays the properties for the selected step in the **Test Steps** pane of a visual test or the result of a visual test or test script. Properties display as name/value pairs. Values for properties can be a literal value, a variable, or an ActiveData value.

You can use the **Properties** pane of a visual test to assign values for the properties of the selected step. This allows you to tailor precise information about how Silk Test Workbench interacts with a test application, manages playback, uses variables, or calls other tests of functions. The **Properties** pane of a result is read-only.

The **Properties** pane contains a toolbar with the following options:

Categorized	Groups properties by property category.
Alphabetical	Lists properties alphabetically.
Description	Displays the help description panel at the bottom of the **Properties** pane.

The **Properties** pane for a result contains an additional toolbar button:

Show/Hide Step Properties of Visual Test Before Playback	Displays the visual test properties of a step as they appeared in the visual test before playback.

Properties that display for a test step are based on the type of step.

Viewing the Properties Pane for Visual Tests

The **Properties** pane lists the properties associated with an asset and the current setting for each property. You can use the **Properties** pane to change an asset's property value. When you select multiple assets or objects, the **Properties** pane contains a list of the properties common to all the selected assets or objects.

To change a property's value, change the property's value in the text box adjacent to the property name. For those properties that have a predefined set of values, select a value from the values displayed in the associated list box.

1. From the **Test Steps** pane, select the step that contains the properties that you want to view and then click **Actions** > **Properties** (or press F4).
2. Click the **Alphabetic** toolbar button to display properties in alphabetic order or click the **Categorized** toolbar button to display properties according to category type.

Visual Navigator: Storyboard

The **Storyboard** pane graphically represents the general structure of the current visual test by displaying text or images in a series of thumbnails. The **Storyboard** pane serves as a visual outline of a visual test and provides a high-level overview of a visual test's flow.

Thumbnails are reduced-sized images that represent screens captured from the test application or text describing a step or series of steps. These thumbnails are associated with test steps that capture an application screen. Text thumbnails represent test steps that do not perform direct interaction with the test application. Examples include the <<Start>> and <<End>> steps, along with steps where other visual tests or scripts are called.

Steps that display in the **Test Steps** pane are separated by a solid dark blue line to indicate a logical grouping. Steps grouped in this manner correspond to each thumbnail in the **Storyboard** pane.

Thumbnail size is relative to the size of the resizable **Visual Navigator** window.

The **Storyboard** pane contains the following informational elements:

Element	Description
Thumbnail images	Graphically represents the flow of a visual test. Images represent captured bitmaps of the test application for the corresponding test steps. To indicate the start and end of the visual test, each Storyboard displays the `<<Start>>` thumbnail as the first step and the `<<End>>` thumbnail as the last step A step or group of steps that do not have a bitmap display descriptive text in the thumbnail. A thumbnail may not contain a bitmap of the test application either because the corresponding step or steps do not interact directly with the test application (such as a step that opens a script), or the test application was not captured in a bitmap. You can customize how Silk Test Workbench captures screens from the test application. For more information, see *Record Output: Test Options*. Click a thumbnail in the Storyboard to make it the active Storyboard item. The steps corresponding to the active Storyboard item highlight in the **Test Steps** pane. The corresponding screen displays in the **Screen Preview**. Property information in the **Properties** pane changes to reflect the active thumbnail as well.
Step numbers	Corresponding step numbers display above each thumbnail. For example, if the numbers 5-8 display above a thumbnail, that thumbnail represents steps 5 through 8 in the current visual test.
Icons	Icons located below provide information about the step or group of steps associated with the thumbnail.  A flagged step.  A flagged step assigned to a user.  A logic or error handling step.  A verification logic step. Additionally, in the Storyboard of the **Result** window, these icons appear:  A passed verification logic step.  A failed verification logic step.  **Note:** If a Storyboard thumbnail contains multiple steps of the same type, only one flag will appear under the thumbnail.

Navigating Thumbnails in the Storyboard

Use the following keyboard actions to navigate the thumbnails in the Storyboard:

- `[right arrow]` or `[down arrow]` to move forward through the thumbnails in the Storyboard.
- `[left arrow]` or `[up arrow]` to move backward through the thumbnails in the Storyboard.
- `Page Up` to return to the first step.
- `Page Down` to return to the last step.
- `Home` or `Ctrl+Home` to move to the `<<Start>>` thumbnail at the beginning of the visual test.
- `End` or `Ctrl+End` to move to the `<<End>>` thumbnail at the conclusion of the visual test.

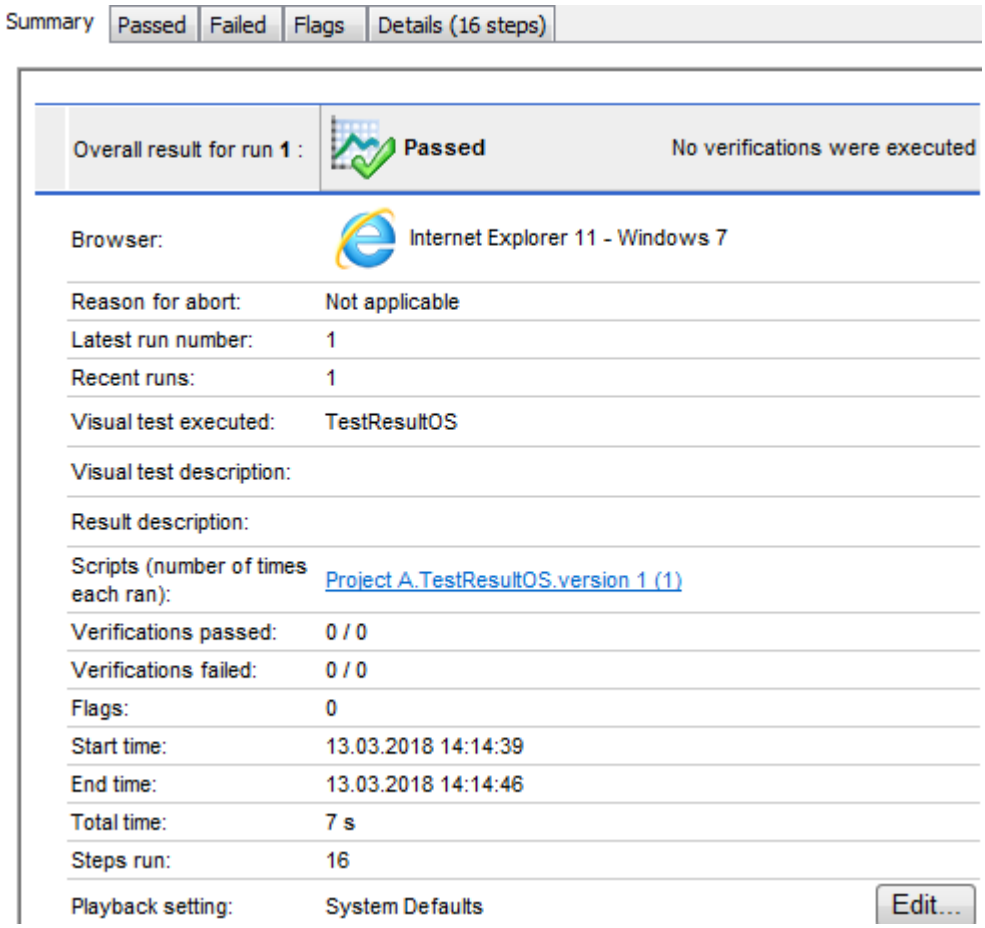
Customizing Visual Navigator Layout

You can dock, hide, or resize the **Screen Preview**, **Properties**, and **Storyboard** panes in the **Visual Navigator**.

1. Each pane and related window in the **Visual Navigator** display a pushpin in the upper-right corner. Click the pushpin to hide a pane. The pane hides and appears as a tab docked to the border of the **Visual Navigator** window.
2. Position your pointer over the tab to temporarily display the pane.
3. Click the tab to temporarily re-dock the pane.
4. When the pane reappears, click the pushpin to permanently re-dock the pane.
5. To adjust the pane size within the **Visual Navigator**, position the cursor between the panes to be resized. The mouse cursor changes to a horizontal or vertical divider. Click and drag the divider to adjust the size of the two panes.
6. Choose **View** > **Restore Layout** to reset **Visual Navigator** layout to the default view.

Result Window

After playing back a visual test or script, you can view the results of the playback in the **Result** window.



The **Result** window contains the following features and functionality:

Visual Navigator	Enables you to quickly see all aspects of test playback. For additional information, see *Visual Navigator in the Result Window*.
Result window toolbar	Enables you to easily customize the display and type of content found in a result. For additional information, see *Result Window Toolbar*.
Result window tabs	The tabs organize result content into specific types. For additional information, see *Result Window Tabs*.

Asset Browser

Use the **Asset Browser** to manage test assets. The **Asset Browser** provides a single point for creating, managing, and viewing assets for each asset type in the database. Press `Ctrl+Alt+A` at any time or click **Asset Browser** from the **Tasks** pane of the **Start Screen** to display the **Asset Browser**.

To view the existing items of an asset type, click the type name under **Asset Types**. A list of related assets displays on the right. To rename, delete, or duplicate assets, right-click an asset and choose the appropriate command. To view asset properties, select an asset and choose **File** > **Summary Information**.

To create a new asset, right-click the asset type and choose **New**. New assets are added to the currently active project.

As tests are developed, a large number of assets may be created. You can modify the view of the assets in several ways so that you can find the items you want.

Controls in the **Asset Browser** include the following:

Active Project	Set the active project. This is the default project to which new test assets are added. When you change the active project, the assets in the new active project are automatically displayed.
View	Check or clear project names to select the projects whose assets you want to view. The assets of the active project are always displayed.
Filter	Filter assets based on set criteria, such as who created or last modified the asset, or when the asset was created or last modified.
Asset Types	Select an asset type from the list to view the test assets available to the selected projects, based on any applied filters. For additional information, see *Asset Types*.
Assets List	Displays the assets of the selected type for the selected projects, based on any applied filters.
Open Item	Type the name of an asset to open.

In addition to using these controls to manage assets in the **Asset Browser**, you can:

- Click a column header in the item listing to sort the assets by that category. Click the header again to reverse the sorting.
- Right-click a column header and choose **Customize Columns** to choose the columns to display or change the column order.
- Enter an item name in the **Open Item** text box, to go directly to the item.

Note: To manage a large number of assets, you can open multiple instances of the **Asset Browser**. The maximum number of **Asset Browser** instances you can open is 32.

Identify Object Dialog Box

You can use the **Identify Object** dialog box to identify the locator of an object in the application under test (AUT), including objects that are not visible, for example containers. Depending on whether you are using object map entries during testing or not, the locator is the name of the object map item or the XPath locator of the selected object. When using object map entries during testing, the name of an object map item is used as an alias of the XPath locator of an object. When you have identified the locator of an object with the **Identify Object** dialog box, you can use the locator in visual tests and VB .NET scripts. You can also use the **Identify Object** dialog box to validate that an XPath locator or an object map entry is valid and corresponds to the expected object.

You can start the **Identify Object** dialog box in several ways, including the following:

- In the toolbar, click **Identify Object**.

  Note: If you have not specified a test application, the **Select Application** dialog box opens, allowing you to specify the application from which you want to identify the object.
- In the menu, click **Tools** > **Identify Object**.

  Note: If you have not specified a test application, the **Select Application** dialog box opens, allowing you to specify the application from which you want to identify the object.
- In the **Start Screen**, select the **Get Started** tab and click **Identify Object**.

  Note: The **Select Application** dialog box opens, allowing you to specify the application from which you want to identify the object.
- In an existing visual test, select a test step that refers to an object that you want to change. Open the **Properties** pane, click into the **Locator** field and select **Identify a Control** > **Identify Object Dialog**.

The **Identify Object** dialog box includes the following controls:

Start Identify	Click **Start Identify** to start recording object information in the test application. Click **Stop Identify** to bring the application under test into the appropriate state before recording a locator.
Selection mode	The selection mode defines how you can select the object that you want to identify. In **Click** mode, click on the object that you want to identify. In **Hot Key** mode, move the mouse cursor over the object that you want to identify and use the keystroke combination specified in the **Keystroke** list box. Typically, you choose the **Hot Key** mode to capture objects that only appear when they are clicked by the user, for example menus or combo-boxes. With the **Hot Key** mode, you can select the object and then press the hot key keystroke combination to capture the locator without dismissing the object.
Selected Locator	Displays the captured locator. When you click **Start Identify**, the **Selected Locator** field displays the locator of the object in the AUT, over which the mouse cursor is currently located. **Note:** If you are testing on a browser, the **Selected Locator** field displays the locator only when you actually capture it.
Test	Click this button to highlight the selected object in the application under test. You can use this button to verify that you have selected the correct object.
Hide Locator Details/Show Locator Details	Click this link to show or hide the **Locator Details** table.
Locator Details table	The **Locator Details** table lists the objects and their related class type, locator value, and attributes for the application that you are testing, which enables you to see the hierarchy of the application. You can use this table to refine your selection and the attributes that you want to assign to the object. For instance, sometimes the auto-generated locator is not the best locator to use in a given test. You can use the **Locator Details** table to find a more relevant property to use with the locator.
Show object maps	Check this check box to display object map item names in the **Locator** column. Object map item names associate a logical name (an alias) with a control or a window, rather than the control or window's locator. By default, object map item names are displayed. To use locators, uncheck this check box.
Show full locators	Check this check box to display the full locator name. To show only the attribute associated with the object, uncheck this check box.

Show properties	Check this check box to display any attributes and attribute values for the object selected in the **Locator Details** table. You can select attributes in this table to use in the locator identification. To hide the Properties subtree and display only locator details, uncheck this check box.
Paste	Click this button to copy the **Selected Locator** into the appropriate visual test, script, or object map.

After you capture an object with the **Identify Object** dialog box, you can:

- Expand the **Locator Details** tree to view the application hierarchy using the context menu.
- Retrieve locators for objects in the subtree using the context menu.
- Show additional locator attributes by clicking the link in the subtree.
- Select an object in the subtree to replace the existing object listed in the **Selected Locator** text box.
- Copy the locator into an existing locator in the **Properties** pane for a visual test.
- Copy the locator directly into a .NET script.

  In the script, you must specify the code to use with the locator. Only the locator is copied.
- Copy the locator into an object map item for an object map.

Code Window

Use the **Code** window to record, design, and modify scripts.

To access the **Code** window, open an existing script or create a script.

Main Page

When you open the **Code** window, the **Main** page is populated with a basic template. After you record a script, the **Main** page contains all the steps that you record. You can manually type code in this page as well. The **Main** page also displays in the **Properties** pane in the **Files** category.

Properties Pane

Use the **Properties** pane to add, view, or modify application configurations, files, parameters, and script references. The **Properties** pane contains the following categories:

- **Application Configurations** – Expand this category to list the application configurations for the script.
- **Files** – Expand this category to display the **Main** page of the **Code** window and any additional files that you have added. The **Main** page contains all the steps that you record.
- **Parameters** – Expand this category to view the input and output parameters for the script. You can add, edit, and delete input and output parameters if necessary.
- **References** – Expand this category to view the references for the script library. You can add and delete references if necessary.

Viewing the Properties Pane for Scripts

Use the **Properties** pane to add, view, or modify application configurations, files, and script references. You can dock, hide, or resize the **Properties** pane in the **Code** window.

1. Open a script. The **Code** window opens.
2. The **Properties** pane displays a pushpin in the upper-right corner. Click the pushpin to hide the pane. The pane hides and appears as a tab docked to the border of the **Code** window.
3. Position your pointer over the tab to temporarily display the pane.
4. Click the tab to temporarily re-dock the pane.

5. When the pane reappears, click the pushpin to permanently re-dock the pane.
6. To adjust the pane size, position the cursor between the panes to be resized. The mouse cursor changes to a horizontal or vertical divider. Click and drag the divider to adjust the size of the two panes.

Output Window

The **Output** window displays information about errors encountered while playing back a script, results of searches performed on scripts, and any text written by a script.

The **Output** window also displays playback warning messages when an object map entry being referenced is not in scope for the asset being executed.

Note: Only the results of NTF methods (calls), such as `Pushbutton.Select`, are recorded in the log. Therefore, if you execute a VB.NET command, such as a `with` statement, the VB.NET call is not logged nor is the `_desktop.Window()` call.

Includes Window

You can reuse test assets by including them in multiple scripts. Each script must explicitly include other test assets that it uses. Silk Test Workbench automatically creates the inclusion references for non-VB.NET assets but users must explicitly include VB.NET assets in each script that uses them.

You can use the **Includes** window to view include information for a selected asset, both the assets that the selected asset includes, and the assets that include the selected asset.

1. Choose **View** > **Includes**. The **Include Info** window opens.
2. In the **Asset Browser** window, select the asset about which you want Includes information.
3. In the **Include Info** window, click the **Includes** tab to see the assets that the selected asset includes or click the **Included By** tab to see the assets that include the selected asset.

Known Issues

This section identifies any known issues with Silk Test Workbench and their resolutions.

General Issues

Object Map Takes a Long Time to Open

If you have a large object map asset it takes a long time to load when you are using .NET 4. Install .NET 4.5 to resolve this issue.

When a remote desktop or remote desktop connection (RDC) is minimized, Silk Test does not function

When you connect through the remote desktop protocol (RDP) to a desktop, you take ownership of the desktop by attaching to the desktop with your mouse and keyboard. If the desktop is minimized without ownership of the desktop being released, any playback of mouse clicks or keystrokes is undefined.

As a workaround, you could use a VNC-based remote viewing tool. This would allow replay to continue even if the client window is minimized.

The Open Agent does not start when the Check Point firewall is installed

When you have a Check Point firewall or a Check Point ZoneAlarm firewall installed on your system, the Open Agent cannot be started, because the firewall interrupts the communication between the Agent and the infoservice.

To start the Open Agent, you have to uninstall the Check Point firewall from your system.

The `modifiers` parameter in the `domDoubleClick` method is ignored

You cannot specify the modifier in the overloaded `domDoubleClick` method. The modifier will not be double-clicked, although you have specified the parameter. The overloaded `domDoubleClick` method, which allows you to specify the modifier, is deprecated. To specify the modifier, you can use the `doubleClick` method, if you are using a client that supports an overloaded method with the `modifiers` parameter, or the `PressKeys` and `ReleaseKeys` methods.

Silk Test does not support testing Metro-style apps

Silk Test does not support testing Metro-style apps on Microsoft Windows 8, Microsoft Windows 8.1, or Microsoft Windows 10. Metro-style apps are also known as Windows 8 style, Modern UI style, Windows Store style, or Universal Windows Platform (UWP) apps.

The built-in spell checking in Microsoft Windows 8 might interfere with the replay of tests

The built-in spell checking in Microsoft Windows 8 can be enabled in applications like Internet Explorer 10.

If a word was incorrectly spelled during recording, and you replay typing this word, the spell checker will either mark it, or for commonly misspelled words will automatically fix it, which is the same behaviour a real user would get. If your tests were created on an operating system that did not include the spell checking feature, you might get unexpected results when replying the tests on Microsoft Windows 8. To disable the spell checking, you can do the following:

1. Press **Windows Key + C**.

2. On the Charm bar, click **Settings**.
3. Select **More PC Settings**.
4. Select **General** to see the Spelling selections.

   Note: These are system-wide settings, not settings specific to Internet Explorer.
5. Set **Autocorrect misspelled words** to off.
6. Set **Highlight misspelled words** to off.

When a .NET application is started from DevPartner Studio (DPS), Silk Test might not recognize it

To resolve the issue, perform the following steps:

1. Go to the Silk Test installation folder (by default, it's located at: `C:\Program Files\Silk\SilkTest`).
2. For Windows Forms applications, go to `ng\agent\plugins\com.borland.fastxd.techdomain.windowsforms.agent_<version number>`.
3. For Windows Presentation Foundation (WPF) applications, go to `ng\agent\plugins\com.microfocus.silktest.techdomain.wpf.agent_<version number>`.
4. In Notepad, open the file plugin.xml, and add the following line to the `<loadparameters>` section:

   ```
   <param name="frameworkAssembly">mscoree.dll</param>
   ```
5. Log out of the computer, and then log back in. Silk Test works as expected with the application that was started by DevPartner Studio.

The highlighting rectangle is out of place when recording clicks on an area of an image

When you record a click on a part of a complex image, for example an area map, the green highlighting rectangle does not highlight the appropriate area of the image. However, the click will be executed correctly during replay.

The Open Agent might not start if Windows Defender is enabled during the installation of Silk Test

If Windows Defender is enabled on your system during the Silk Test installation, you might not be able to start the Open Agent after the installation is complete. Windows Defender might prevent the hotfix setup from performing some required actions. As a workaround, disable Windows Defender during the Silk Test installation.

The IME editor opens out of place when opened from specific locations

The IME editor opens on the top-left corner of the current screen instead of opening near the text field from which it is opened.

This behavior occurs when opening the IME editor from the following locations:

- The **Silk Recorder**.
- The **Keyword-Driven Test Editor**.
- The **Keywords** window.

Silk Test help freezes when you enter characters in the Search box

The Silk Test help might freeze when you enter characters in the **Search** box on the **Index** tab in a Compiled HTML Help (.chm) file in Microsoft Windows 8.1, Microsoft Windows RT 8.1, or Microsoft Windows Server 2012 R2. To fix this issue, you have to install a Windows update. The update is available from *https://support.microsoft.com/en-us/kb/3080042*.

Cannot use Shift+Insert when the Numeric Lock key is active

Using the `TypeKeys` method to paste the contents of the clipboard with **Left Shift+Insert** or **Right Shift +Insert** does not work when the Numeric Lock (Num Lock) key is active.

You can workaround this issue by deactivating the Num Lock in your test script before calling the TypeKeys method.

Mobile Web Applications

Silk Test Workbench does not support frames on Apple Safari

Silk Test Workbench does not support HTML frames and iframes on Apple Safari on iOS.

Chrome for Android 43 or later: Silk Test Workbench does not support zooming and scrolling at the same time

Recording a mobile web application on Chrome for Android 43 or later, while the mobile web application is zoomed and the top-left corner is not visible on the screen, might not work as expected. If the green rectangles for the controls in a mobile web application are not correctly displayed during recording, zoom-out the mobile web application completely, scroll to the top-left corner of the mobile web application, and refresh the **Recording** window.

Web Applications

Recording with a zoom level different to 100% might not work properly

Recording a Web application with a zoom level different to 100% might not work as expected. Before recording actions against a Web application, set the zoom level in the browser to 100%.

Google Chrome

Locator recording in windows fails with Google Chrome

When you are testing a Web application in Google Chrome, locator recording in windows fails when multiple windows are open during application configuration in the Google Chrome instance, in which the application is running. If you close the other Google Chrome windows during application configuration, the error will not appear.

Background applications in Google Chrome prevent automation support from loading

When you want to test a Web application with Google Chrome and the **Continue running background apps when Google Chrome is closed** check box is checked, Silk Test cannot restart Google Chrome to load the automation support.

Silk Test cannot record locators in modal dialogs when Windows Aero is disabled

If Windows Aero functionality is disabled, modal dialogs are not recognized and locators in such dialogs cannot be selected. As a workaround, use the **Locator Spy** or the **Identify Object** dialog box to manually create and verify locators while a modal dialog is displayed.

Silk Test does not display embedded PDFs

With Google Chrome 42 or later, Google Chrome by default blocks the NPAPI plug-in, which is used to display embedded PDFs. Because of this, Silk Test does not display embedded PDFs in Google Chrome 42 or later, but instead downloads the embedded PDFs.

- If you are using Google Chrome 44 or prior, you can unblock the NPAPI plug-in in Google Chrome, by typing the following into the address bar:
  ```
  chrome://flags/#enable-npapi
  ```
- If you are using Google Chrome 45 or later, the NPAPI plug-in is completely removed from Google Chrome, without an option to re-enable it, and all PDFs are downloaded.

Connection timeouts when executing tests against Google Chrome 49 or prior

When executing tests against Google Chrome 49 or prior on a slow machine, you might experience connection timeouts, causing the tests to fail. The following error message is displayed:
Error executing '*'. Communication with browser automation timed out.

To avoid such connection timeouts, ensure that the test machine has enough processing power. For example, if you are testing on a slow virtual machine (VM), you could enhance the processing power by adding an additional CPU core to the VM.

Internet Explorer

Using Google toolbar interferes with recording Web applications

Using the Google toolbar with Internet Explorer 8 interferes with recording locators for Web applications.

Turn off the Google toolbar before you record Web applications.

Microsoft Silverlight Applications

Some Microsoft Silverlight applications cause Internet Explorer to hang when interacting with Silk Test. On 32-bit platforms, refer to MS KB 2564958 (an update to Active Accessibility) to help prevent the issue.

Locators recorded with Silk Test versions prior to Silk Test 13.5 might no longer work in Internet Explorer

In Silk Test 13.5, we have adapted the normalization of white spaces of the `textContents` attribute in Internet Explorer. This change was made to improve the cross-browser capabilities of Silk Test, and might affect locators which rely on the `textContents` attribute, and which are used in scripts that were recorded with releases prior to Silk Test 13.5.

The Open Agent cannot have high elevation enabled when UAC is enabled on Microsoft Windows 8 or later and Internet Explorer 11

You cannot test a Web application in Internet Explorer 11 on Microsoft Windows 8 or later and have UAC enabled and run both Internet Explorer and the Open Agent with high elevation.

Known issues with Input Method Editors (IMEs)

- Silk Test does not record half-width spaces for Japanese input in Internet Explorer 11.
- Silk Test does not record IME input in Internet Explorer 11 in compatibility mode.
- In Japanese IME mode, pressing **Space** will cause Silk Test to record the current IME candidate. Use **Convert** to avoid this issue.

Internet Explorer might stop working while testing an Applet with a Java version higher that 1.7 update 71

Internet Explorer might stop working while testing an Applet with a Java version higher that 1.7 update 71 (7u71).

Internet Explorer does not respond during test execution

Internet Explorer 10 or later might become unresponsive during test execution because a thread in Internet Explorer is suspended and creates a deadlock. The root cause of this problem is a new security feature in Internet Explorer.

To solve this issue, disable the security feature.

1. Open the **Registry Editor**.
2. If the **OverrideMemoryProtectionSetting** entry does not exist in the key **HKEY_CURRENT_USER \SOFTWARE\Microsoft\Internet Explorer\Main**, create the entry.
3. Set the value of the registry key **HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer \Main::OverrideMemoryProtectionSetting** to 2.

The JavaScript alert-handling API methods do not work with an embedded Internet Explorer

The following JavaScript alert-handling methods of the `BrowserWindow` class do not work when testing an embedded Internet Explorer:

- `AcceptAlert` method
- `DismissAlert` method
- `GetAlertText` method
- `IsAlertPresent` method

Microsoft Edge

Windows opened out of Microsoft Edge are not supported

Windows which are opened as actual new Microsoft Edge windows, and not as tabs, are not supported. Silk Test Workbench cannot close such windows correctly, and the agent is in an invalid state after closing such a window.

JavaScript alerts opened in an iframe or frame cannot be closed

Microsoft Edge cannot close the JavaScript alerts `alert`(), `prompt`(), and `confirm`(), if these alerts were opened in an iframe or frame.

Cannot open the native browser context menu

If a test includes a right click in Microsoft Edge, which opens the native browser context menu, the test will hang. This issue does not occur with HTML menus opened by Microsoft Edge.

Cannot execute tests against Microsoft Edge with UAC disabled

When executing tests against Microsoft Edge with UAC disabled, the following error message displays: Failed to start application 'Edge'. Unable to parse remote response: Unknown error

To solve this issue, enable UAC.

Mozilla Firefox

Calls into applications using Adobe FlashPlayer do not properly synchronize when using Mozilla Firefox

When you are using Mozilla Firefox with a recent Adobe FlashPlayer version, some calls might not synchronize properly. The following issues might occur:

- Mozilla Firefox might falsely recognize a running script as stalled and a confirmation dialog box might appear asking whether you want to continue the execution of the script, even though the script is running properly.
- Typing characters might not work because `SetFocus` is no longer working correctly.
- The Adobe automation might return an old value although the UI already shows a new value.

If you face one or more of these issues with applications using Adobe FlashPlayer, turn off the ProtectedMode in Adobe FlashPlayer. For additional information, see *http://forums.adobe.com/thread/1018071* and read the information provided under *Last Resort*.

SAP Applications

`HierarchyHeaderWidth` and `ColumnOrder` Properties of the SAPTree Class are write only

Other than the automation documentation indicates, the `HierarchyHeaderWidth` and `ColumnOrder` properties of the `SAPTree` class are write only and cannot be read.

Ensure that your scripts use write rather than read with these properties.

`GetColumnIndexFromName()` of the SapTree Class May Fail with an "unspecified error"

`GetColumnIndexFromName()` of the `SapTree` class may fail with an "unspecified error". This is a known issue in SAP automation.

Check the SAP web site to see if the issue has been resolved.

Calling the `Select()` method of the SAPTree Class on a Context menu item may fail

Calling the `Select()` method of the `SAPTree` class on a Context menu item may fail.

Call `SelectContextMenuItem` on the parent control instead. This problem is a known issue in the SAP automation.

The position property for a horizontal scrollbar always returns 1

The position property for a horizontal scrollbar always returns 1. This is a known issue in SAP automation.

Check the SAP web site to see if the issue has been resolved.

The SAPNetPlan class is not supported

This issue will be resolved in a future release.

Replay error occurs when executing an SAP script

In certain cases, if you record an SAP test and then replay it, the following error might occur: `The data necessary to complete this operation is not yet available.` This means that Silk Test Workbench is executing the recorded actions too fast.

To solve this issue, you can add sleeps to your test script or you can increase the post replay delay to increase the time that Silk Test Workbench waits after invoking a function or setting a property. For additional information about this option, see *Agent Options*. You could also change the script to use SAP automation to replay the problematic action instead of using the xBrowser technology domain. For example, you could change the action `DomLink.Select` to `SapHTMLViewer.SapEvent`.

The method getCurrentRow returns a wrong value with SAPGUI client 7.30

If you use SAPGUI client 7.30 and you call the method `getCurrentRow`, the method might falsely return -1 instead of the row number.

The method resizeWorkingPane is not working correctly with SAPGUI client 7.30

If you use SAPGUI client 7.30 and you call the method `resizeWorkingPaneEx`, the method will not resize the `workingPane` and calling `getSapWindow().getWidth()` will return a wrong value for the window width.

Cannot start recording on a virtual machine when Silk Test Workbench is started from eCATT

When you have created an eCATT script from SAPGui, recording actions on a virtual machine by pressing the **Record** button, while the VB .NET script, visual test, or keyword-driven test is open in the Silk Test Workbench editor, does not work. Recording actions on a physical machine works.

When importing assets to a Silk Test Workbench database from eCATT, no error is displayed when using an eCATT BLOB that was created with a newer version of Silk Test Workbench than the XML of the exported assets

When using a importing assets from eCATT to a Silk Test Workbench database, the importing process fails silently if the version of Silk Test Workbench is lower than the version of the XML, which includes the assets, in the eCATT BLOB.

If the import fails, the next time that you try to edit the blob, the following message displays:
Would you like to save the current BLOB before returning to eCATT?
In this case, do not save the BLOB, because it is empty. Instead, click **No** to return to eCATT.

Tip: If you are using multiple instances of Silk Test Workbench with eCATT, Micro Focus recommends updating all instances to the same Silk Test Workbench version.

Oracle Forms

Silk Test Workbench does not support testing Oracle Forms with a Java version higher than 1.7 update 60

Silk Test Workbench does not support testing Oracle Forms with a Java version higher than 1.7 update 60 (7u60).

Silk Test Workbench

Replaying a Visual Test on a Large Site

You may encounter performance problems when running a visual test on a large site when using xBrowser. In order to avoid this, set the option **Playback** > **Results** > **Visual test** > **Control capture** to `No`.

Message boxes display in the background when a script plays back (31314)

If you include a message box statement, such as `MsgBox ("Hello")` in .NET script, it displays in the background when you play back the script.

Include `MsgBoxStyle.MsgBoxSetForeground` in the `MsgBox` statement to have the message box display in the foreground. For example:

```
MsgBox ("Hello", MsgBoxStyle.MsgBoxSetForeground)
```

Installer fails to install SQL Server Native Client and register Silk Test Workbench as a COM server

If the operating system performs an automatic update or the user initiates a Windows update during or before installation of Silk Test, the installer fails to install SQL Server Native Client and register Silk Test Workbench as a COM server.

Both of these problems affect Silk Central Test Manager (SCTM) integration, since SCTM uses the native client driver installed by SQL Server Express to create Silk Test Workbench DSNs. Furthermore, SCTM needs Silk Test Workbench registered as a COM server in order to use its COM interface for automation.

Choose one of the following solutions:

1. Do not install Silk Test while running a Windows update.
2. Ensure that Windows updates do not install automatically on your machine.
3. Install any pending Windows updates prior to installing Silk Test and reboot the system before installing Silk Test.

Modifying the hot key combination to include multiple modifiers causes additional keys to be recorded

If you configure a hot key for insert verification or start/stop record that contains multiple modifiers, occasionally the modifier press key action is recorded. For instance, if you specify a hot key combination of `Alt+Ctrl+F9`, `Alt` may be recorded as a press key action instead of being ignored as part of the hot key combination. This is a problem during playback because the appropriate release key actions are missing.

Manually remove the actions related to the modifiers, or only use one hot key modifier.

Data Source Names (DSNs) do not work on 64-bit systems

A 64-bit DSN for SQL Server or Oracle cannot be used with Silk Test Workbench. To create a DSN for a 64-bit machine, click **Start** > **Silk** > **Silk Test** > **Administration** > **Data Sources (ODBC)** and create a 32-bit DSN. You can also use the WOW64 tools located at `C:\WINDOWS\SysWOW64\odbcad32.exe`.

Silk Test Workbench requires a network adapter to start

A network adapter must be available on the machine on which Silk Test Workbench is installed to enable the communication between Silk Test Workbench and the Open Agent during the start of Silk Test Workbench. If no network adapter is available, a `"Failed to connect to OpenAgent"` error occurs. Other clients, such as Silk Test Classic and Silk4J, do not require a network connection to start.

Ensure that a network adapter is available before starting Silk Test Workbench.

SQL Server Express might not install on Windows Vista SP1 machines

On Windows Vista SP1 machines, ensure that the full version of .NET 3.5 SP1 is installed if you plan to install SQL Server 2008 Express. To install the full version of .NET 3.5 SP1, download it from the Microsoft Web site.

Identify from Screen Preview creates an incorrect object map item for a migrated database

After a database is migrated from Silk Test Workbench 2010 to Silk Test Workbench 2010 R2, using the **Identify from Screen Preview** creates incorrect object map items.

Use the **Update Screen** command to recapture the screen before using **Identify from Screen Preview**.

When a child script is called from a parent script, the child script can access the object map out of scope

If a parent and child script reside in different projects and use different object maps, the child script runs successfully even though the object map is out of scope for the child script.

Since the script runs successfully, there is no resolution necessary. However, this behavior will not work in future releases.

When multiple users attempt to simultaneously edit the same asset on a SQL 2008 database, issues with the asset locking mechanism might occur

To test for the problem, open an asset, like a visual test, for editing in the Silk Test Workbench. While the asset is open, attempt to open the same asset from a different instance of Silk Test Workbench. If the problem occurs, either the second instance of Silk Test Workbench will be able to simultaneously edit the same asset, or the second instance will receive a message with incomplete data. The user name and machine name will both be blank.

In SQL Server, allocate database admin rights to the SQL database users that use that database.

Unexpected error while exporting assets in a SQL database

The unexpected error, `Conversion failed when converting from Character string to Unique identifier`, occurs when exporting assets in a SQL database.

Import the exported database and press OK if any error messages occur. The import finishes successfully and the assets are available for use.

Unexpected behavior while using an Access database

Compact the database using the Silk Test **Database Maintenance** tool. Click (in Microsoft Windows 7) **Start** > **Programs** > **Silk** > **Silk Test** > **Administration** > **Database Maintenance** or (in Microsoft Windows 10) **Start** > **Silk** > **Database Maintenance**. Then, open the database and click **Tools** > **Compact Database**.

Silk Test Workbench with UAC enabled during configuration cannot connect to an SQL Server

You cannot connect to an SQL server during configuration, when you are using Silk Test Workbench without administrator rights, and with UAC enabled.

To connect to the SQL server during configuration, you have to start Silk Test Workbench with the **Run as administrator** option.

Silk Test Workbench might crash when the connection to a remote SQL database is lost

When you are using a remote SQL database, and that database disconnects from Silk Test Workbench, for example because of a server restart, Silk Test Workbench might crash when trying to communicate with the database.

To reconnect with the database, restart Silk Test Workbench.

The replay of a visual test might get very slow if the Control capture option is set

Set the Control capture option to **No** to improve playback performance.

Microsoft Windows 7 Classic theme: Screen Previews of visual tests might contain the Recording window

Screen previews of visual tests which are recorded on a Microsoft Windows 7 machine with the Classic theme might contain the **Recording** window. To record visual tests in Microsoft Windows 7 without the **Recording** window, do not use the Classic theme.

Renaming an asset without updating references

When renaming assets that are referenced by other assets, ensure to also update the references. If you do not update the references directly, you cannot simply rename the asset back to the original name and then update the references by renaming the asset again, because Silk Test Workbench will not prompt you to update the references.

If the asset that you want to rename is a .NET script, and the referencing asset is a visual test from which the .NET script is called, you can workaround this issue by saving the visual test. This will cause Silk Test Workbench to find the .NET script again and to set everything up correctly to update asset references.

Silk Test Workbench Language Reference

This section lists the classes, methods, and properties for each technology domain that Silk Test Workbench supports.

The scripting language for Silk Test Workbench is Visual Basic .NET.

For additional information on the native classes, methods, and properties of Visual Basic .NET, refer to *http://msdn.microsoft.com/en-us/library/2x7h1hfk*.

Common Class Reference

Lists the available classes for testing Win32, Java SWT, and Windows Forms controls.

BaseGuiTestObject Class

Description
The base class for GUI objects that can have a context menu.

Inheritance Hierarchy

- *GuiTestObject*
  - BaseGuiTestObject
    - *AccessibleControl*
    - *Control*
    - *Window*

Syntax

```
'Declaration
Public Class BaseGuiTestObject _
Inherits GuiTestObject _
Implements IScrollable
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

CharSet Enumeration

Description

Specifies the character set to use when reading files.

Inheritance Hierarchy

- *SystemFunctions*
  - CharSet

Syntax

```
Public Enumeration CharSet {
 Undefined = 100,
 ANSI = 101,
 UTF8 = 102,
 Unicode = 103
}
```

Members

Name	Description
Undefined	No character set is specified.
ANSI	The ANSI character set is used.
UTF8	The UTF8 character set is used.
Unicode	The Unicode character set is used.

CheckBox Class

Description
The class for check box controls.

Inheritance Hierarchy

- *Control*
  - CheckBox

Syntax

```
'Declaration
Public Class CheckBox _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)

Name	Description
State	The state of a check box. Values include: 1=checked, 2=unchecked, 3=undecided.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsChecked	Returns *true* if the checkbox is checked and *false* otherwise.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Select	Selects a check box.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetUndecided	Sets the check box to the undecided state.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Toggle	Toggles the state of the check box. If the checkbox is checked it is unchecked and vice versa. If the checkbox is in the undecided state it is left undecided.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

CheckBoxToolItem Class

Description
The class for a check box element in a toolbar control.

Inheritance Hierarchy

- *ToolItem*
  - CheckBoxToolItem

Syntax
```
'Declaration
Public Class CheckBoxToolItem _
Inherits ToolItem
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
State	The state of a check box. Values include: 1=checked,2=unchecked, 3=undecided
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by

Name	Description
	the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ComboBox Class

Description

The class for controls that have both a popup list and a text field. If the user selects an item from the list, the text field is filled with that string. Alternatively, the user can type the string into the text field.

Inheritance Hierarchy

- *Control*
  - ComboBox

Syntax

```
'Declaration
Public Class ComboBox _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)

Name	Description
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the combo box list.
Items	A list of items in the combo box list.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The index of the selected item.
SelectedItem	The selected item.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearText	Removes all text from the combo box.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item from the combo box.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetText	Replaces the text in the text field of the combo box.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Control Class

Description
The class for any control in a window.

Inheritance Hierarchy

- *BaseGuiTestObject*
  - Control
    - *CBanner*
    - *CheckBox*
    - *ComboBox*
    - *CoolBar*
    - *CTabFolder*
    - *DataGrid*
    - *DomainUpDown*
    - *ElementHost*
    - *ExpandBar*
    - *FormsHost*
    - *Group*
    - *Header*
    - *Label*
    - *Link*
    - *ListBox*
    - *ListView*
    - *MenuStrip*
    - *MonthCalendar*
    - *Pager*
    - *ProgressBar*
    - *PushButton*
    - *RadioList*
    - *Sash*
    - *SashForm*
    - *Scale*
    - *ScrollableControl*
    - *ScrollBar*
    - *ScrolledComposite*

- *Spinner*
- *StatusBar*
- *SWTBrowser*
- *SWTDateTime*
- *TabControl*
- *Table*
- *TextField*
- *ToggleButton*
- *ToolBar*
- *Tree*
- *UpDown*
- *ViewForm*

Syntax

```
'Declaration
Public Class Control _
Inherits BaseGuiTestObject
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SystemFunctions Class

Description

Allows clients to perform operations on the remote Open Agent host.

Inheritance Hierarchy

- `SystemFunctions`
  - *CharSet*
  - *ExecutionMode*
  - *ExecutionResult*
  - *FileHandle*
  - *FileInfo*
  - *FileOpenMode*
  - *FilePointerMode*
  - *FileShareMode*
  - *FileSizeUnit*
  - *RegistryCategory*
  - *RegistryView*

Syntax

```
'Declaration
Public Class SystemFunctions _
```

Properties

Name	Description
AgentDirectory	Gets the directory from which the Open Agent is running.
AgentVersion	Gets the version of the Open Agent.
ClipboardText	Gets or sets the text in the clipboard.
CurrentDrive	Gets or sets the current drive of the machine on which the Open Agent is hosted.
CurrentRegistryView	Gets or sets the current registry view.
CurrentWorkingDirectory	Gets or sets the current working directory of the Open Agent.
CursorPosition	Gets the current cursor position of the machine on which the Open Agent is running.
CursorType	Gets the current cursor type of the machine on which the Open Agent is running.
FreeMemory	Gets the percentage of free memory that is available on the machine on which the Open Agent is hosted.
HostName	Gets the hostname of the machine on which the Open Agent is hosted.
Locale	Gets the locale of the machine on which the Open Agent is hosted, for example en-US.
OSName	Gets the name of the operating system of the machine on which the Open Agent is hosted.
OSVersionType	Gets the version type, server or client, of the operating system of the machine on which the Open Agent is hosted.
OSVersion	Gets the major and minor version of the operating system of the machine on which the Open Agent is hosted.

Methods

Name	Description
CompareBinaryFiles	Performs a binary comparison on two files.
CompareBitmaps	Compares two bitmaps.
CompareTextFiles	Compares two text files lexicographically.
CopyFile	Copies a file.
CreateDirectory	Creates a new directory with the specified name.
CreateRegistryKey	Creates a new key in the registry. Recursively creates any missing registry keys in the given keyPath.
CreateRegistryValue	Creates a new value.
DeleteRegistryKey	Deletes a key from the registry.
DeleteRegistryValue	Deletes a value.
DirectoryExists	Checks whether the specified directory exists.

Name	Description
Execute	Executes the specified command on the machine on which the Open Agent is hosted.
ExistsRegistryKey	Checks whether the specified key exists in the registry.
ExistsRegistryValue	Checks if a value exists in the registry.
FileClose	Closes a previously opened file.
FileExists	Checks whether the specified file exists.
FileOpen	Opens a file. If the file does not exist, it is created.
FileReadLine	Reads a single line from a file. Starts at the current position of the pointer in the file.
FileSetPointer	Sets the position pointer within the file for read and write operations.
FileWriteLine	Writes a line at the current position of the pointer in a file.
GetBitmapCRC	Returns the directory from which the Open Agent is running.
GetDirectoryContents	Gets the contents of a directory.
GetEnvironmentVariable	Gets the value of the specified environment variable.
GetFileInfo	Gets information about the specified file.
GetFreeDiskSpace	Returns how much free disk space is available on the specified drive.
GetRegistryKeyNames	Returns the names of all registry keys below the specified key.
GetRegistryValueNames	Returns the values of all registry keys below the specified key.
GetRegistryValue	Returns the value of the specified registry key.
MoveFile	Moves a file.
RemoveDirectory	Deletes the specified directory.
RemoveFile	Deletes the specified file.
SetEnvironmentVariable	Sets the value of the specified environment variable.
SetRegistryValue	Sets the value of the specified registry key.

Desktop Class

Description

`Desktop` is the class for the entire screen.

Inheritance Hierarchy

- *TestObject*
  - `Desktop`

Properties

`Desktop` inherits all its properties from *TestObject*.

Methods

`Desktop` inherits all its methods from *`TestObject`*.

Dialog Class

Description
The class for dialogs.

Inheritance Hierarchy

- *`Window`*
  - `Dialog`

Syntax

```
'Declaration
Public Class Dialog _
Inherits Window
```

Properties

Name	Description
`Application`	The name of the Application that this Window belongs to. (Inherited from *`Window`*)
`Background`	The background color of the GUI object. (Inherited from *`GuiTestObject`*)
`Enabled`	Whether the GUI object is enabled. (Inherited from *`GuiTestObject`*)
`Font`	The font type of the GUI object. (Inherited from *`GuiTestObject`*)
`Foreground`	The foreground color of the GUI object. (Inherited from *`GuiTestObject`*)
`NativeHandle`	The native window handle for the object. (Inherited from *`INativeWindow`*)
`Text`	The text of the control. (Inherited from *`TestObject`*)
`Value`	The value of the control, e.g.: text in a text control. (Inherited from *`TestObject`*)
`Visible`	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *`GuiTestObject`*)
`WindowState`	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *`IMoveable`*)

Methods

Name	Description
`CaptureBitmap`	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DropDownToolItem Class

Description
The class for drop-down element in a toolbar control.

Inheritance Hierarchy

- *ToolItem*
  - DropDownToolItem

Syntax
```
'Declaration
Public Class DropDownToolItem _
Inherits ToolItem
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Open	Opens an item.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ExecutionMode Enumeration

Description

Defines whether a function waits for the call to finish before executing a command.

Inheritance Hierarchy

- *SystemFunctions*
  - `ExecutionMode`

Syntax

```
Public Enum ExecutionMode
```

Members

Name	Description
WaitUntilFinished	Execute should return only after the program is finished.
ContinueRunning	Start the executable and immediately return. `ExecutionResult`.ExitCode will always be 0, and no output will be captured in `ExecutionResult`.RawOutput.

ExecutionResult Class

Description

Encapsulates the results of a call to `SystemFunctions.Execute`.

Inheritance Hierarchy

- *SystemFunctions*
  - ExecutionResult

Methods

Name	Description
GetExitCode	Returns the result of a call to *Execute*. 0 if no error occurred.
GetRawOutput	Returns the contents of stdout and stderr.
GetOutput	Returns the contents of stdout and stderr as a string.
ToString	Returns the exit code as a string.

FileHandle Class

Description

A logical file handle.

Inheritance Hierarchy

- *SystemFunctions*
  - FileHandle

Methods

Name	Description
GetHandle	Returns the internal value for the handle.
GetName	Returns the name of the file or directory to which the handle corresponds.

FileInfo Class

Description

Structure storing information about a directory entry.

Inheritance Hierarchy

- *SystemFunctions*
  - FileInfo

Properties

Name	Description
Attributes	Gets the attributes of the file.
CreationDate	Gets the date at which the file or directory was created.
IsDirectory	Gets whether the instance stores information about a directory or not.

Name	Description
ModificationTime	Gets the time at which the file or directory was last modified.
Name	Gets the name of the file.
Size	Gets the size of the file.

FileOpenMode Enumeration

Description

Defines how to open a file. The file mode is a required parameter for the *FileOpen* function.

Inheritance Hierarchy

- *SystemFunctions*
  - FileOpenMode

Syntax

```
Public Enum FileOpenMode
```

Members

Name	Description
Read	Opens the file for reading.
Write	Opens the file for writing. If the file does not exist, it is created. If the file exists, it is truncated to zero (0) bytes.
ReadWrite	Opens the file for reading and writing.
Update	Opens the file for writing, but does not truncate the file. Denies write access to other users by default. The file pointer is positioned at the beginning of the file.

FilePointerMode Enumeration

Description

Specifies the position to use when reading or writing a file.

Inheritance Hierarchy

- *SystemFunctions*
  - FilePointerMode

Syntax

```
Public Enum FilePointerMode
```

Members

Name	Description
Start	Sets the pointer to the start of the file.

Name	Description
Relative	Leaves the pointer in the current position in the file.
End	Sets the pointer to the end of the file.

FileShareMode Enumeration

Description

Defines how other programs can access files while they are opened through Silk Test.

Inheritance Hierarchy

- *SystemFunctions*
  - FileShareMode

Syntax

```
Public Enum FileShareMode
```

Members

Name	Description
DenyNone	Other programs can read from and write to the file at the same time as the Open Agent.
DenyWrite	Other programs can read from but not write to the file at the same time as the Open Agent.
DenyAll	Other programs cannot access the file at the same time as the Open Agent.

FileSizeUnit Enumeration

Description

Defines which unit to use when calculating file sizes.

Inheritance Hierarchy

- *SystemFunctions*
  - FileSizeUnit

Syntax

```
Public Enum FileSizeUnit
```

Members

Name	Description
Byte	1 Byte.
KiloByte	2^10 Byte.
MegaByte	2^20 Byte.
GigaByte	2^30 Byte.
TeraByte	2^40 Byte.

Group Class

Description
The class used to combine controls to give the user visual hints that those controls belong together. Consequently, the Group control does not have any other functionality that can be expressed in methods.

Inheritance Hierarchy

- *Control*
  - Group

Syntax

```
'Declaration
Public Class Group _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

GuiTestObject Class

Description
The base class for all GUI objects.

Inheritance Hierarchy

- *TestObject*
  - GuiTestObject
    - *BaseGuiTestObject*
    - *Item*
    - *Menu*

Syntax

```
'Declaration
Public Class GuiTestObject _
Inherits TestObject _
Implements IClickable, IFocusable, IKeyable, INativeWindow
```

Properties

Name	Description
Background	The background color of the GUI object.
Enabled	Whether the GUI object is enabled.
Font	The font type of the GUI object.

Name	Description
Foreground	The foreground color of the GUI object.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

HorizontalScrollBar Class

Description
The class for horizontal scroll bars.

Inheritance Hierarchy

- *ScrollBar*
  - HorizontalScrollBar

Syntax

```
'Declaration
Public Class HorizontalScrollBar _
Inherits ScrollBar
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	The internal page size of the scroll bar. (Inherited from *ScrollBar*)
Position	The actual position of the scroll bar caret. (Inherited from *ScrollBar*)
Range	The range of the scroll bar. (Inherited from *ScrollBar*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PageLeft	Scrolls the scroll bar page-wise left.
PageRight	Scrolls the scroll bar page-wise right.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Scrolls the scroll bar. (Inherited from *ScrollBar*)
ScrollLeft	Scrolls the scroll bar to the left.
ScrollRight	Scrolls the scroll bar to the right.
ScrollToMax	Scrolls the scroll bar to its maximum position. (Inherited from *ScrollBar*)
ScrollToMin	Scrolls the scroll bar to its minimum position. (Inherited from *ScrollBar*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

IClickable Interface

Description
Interface for objects that can be clicked.

Syntax
```
'Declaration
Public Class IClickable
```

Methods

Name	Description
Click	Clicks on the object.
DoubleClick	Double-clicks a mouse button on the object.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications.
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications.

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications.

IFocusable Interface

Description
Interface for objects that can be focused on.

Syntax
```
'Declaration
Public Class IFocusable
```

Methods

Name	Description
IsFocused	Return whether the control has focus.
SetFocus	Gives focus to the control.

IKeyable Interface

Description
Interface for objects that use keystrokes. All methods and properties in this interface are not supported for mobile Web applications.

Syntax
```
'Declaration
Public Class IKeyable
```

Methods

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons.
ReleaseKeys	Releases a set of keys or mouse buttons.
TypeKeys	Sends a set of keystrokes to the object.
TypePasswordKeys	Types an encrypted password into an object, for example a text field.

IMoveable Interface

Description
Interface for objects that can be moved, e.g. windows.

Syntax
```
'Declaration
Public Class IMoveable
```

Properties

Name	Description
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored

Methods

Name	Description
Close	Closes the window.
CloseSynchron	Closes the window and waits until the window is closed.
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open.
GetFocus	Returns the object with the input focus.
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window.
IsActive	Returns whether the window is set to active.
Maximize	Maximizes the window.
Minimize	Reduces the window to an icon.
Move	Moves the window.
Restore	Restores the window to its previous size.
SetActive	Makes the window active.
Size	Resizes the window.

INativeWindow Interface

Description
Interface for objects that provide a native window handle.

Syntax
```
'Declaration
Public Class INativeWindow
```

Properties

Name	Description
NativeHandle	The native window handle for the object.

IScrollable Interface

Description
Interface for objects that can have scroll bars.

Syntax
```
'Declaration
Public Class IScrollable
```

Methods

Name	Description
GetHorizontalBar	Returns the horizontal scroll bar for this control.
GetVerticalBar	Returns the vertical scroll bar for this control.

Item Class

Description
Item base class.

Inheritance Hierarchy

- *GuiTestObject*
  - Item
    - *CoolItem*
    - *CTabItem*
    - *DataGridColumn*
    - *DataGridItem*
    - *DataGridRow*
    - *ExpandItem*
    - *MenuItem*
    - *SeparatorItem*
    - *SWTTabItem*
    - *SWTTreeColumn*
    - *TableColumn*
    - *TableRow*
    - *ToolItem*

Syntax

```
'Declaration
Public Class Item _
Inherits GuiTestObject
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Label Class

Description
The class for static text strings (that is, text that the user cannot change, such as a label).

Inheritance Hierarchy

- *Control*
  - Label

Syntax
```
'Declaration
Public Class Label _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Link Class

Description
The class that opens the topic location referenced in the source.

Inheritance Hierarchy

- *Control*
  - Link

Syntax

```
'Declaration
Public Class Link _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ListBox Class

Description
The class for list boxes.

Inheritance Hierarchy

- *Control*
  - ListBox
    - *CheckedListBox*

Syntax

```
'Declaration
Public Class ListBox _
Inherits Control
```

Properties

Name	Description
AllowsMultiSelect	Whether the control supports selecting multiple items.
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the list box.
Items	A list of items in the list box.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The index of the first selected item.
SelectedIndices	The indices of the selected item(s).
SelectedItem	The name of the first selected item.
SelectedItems	The names of the selected item(s).
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Double-clicks an item in the list box.

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
ExtendSelect	Selects a range of items by extending the selection in the list box.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MultiSelect	Selects an item in the multi- or extend-selection list box.
MultiUnselect	Unselects an item in the multi- or extend-selection list box.
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item from the list box.
SelectRange	Selects a range of items in the extend-selection list box.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by

Name	Description
	the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Menu Class

Description
The class for menus.

Inheritance Hierarchy

- *GuiTestObject*
  - Menu

Syntax

```
'Declaration
Public Class Menu _
Inherits GuiTestObject
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)

Name	Description
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetItemCount	Returns the number of menu items for this menu. Also includes separator menu items.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects a menu.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

MenuItem Class

Description
The class for items on a menu.

Inheritance Hierarchy

- *Item*
  - MenuItem

Syntax
```
'Declaration
Public Class MenuItem _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Checked	Whether the menu item is checked.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item from the menu.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

PushButton Class

Description
The class for buttons.

Inheritance Hierarchy

- *Control*
  - PushButton

Syntax

```
'Declaration
Public Class PushButton _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)

Name	Description
Pressed	Whether the pushbutton is pressed.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects a pushbutton.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

PushToolItem Class

Description
The class for a pushbutton element in a toolbar control.

Inheritance Hierarchy

- *ToolItem*
  - PushToolItem

Syntax
```
'Declaration
Public Class PushToolItem _
Inherits ToolItem
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

RadioList Class

Description

The class for a group of radio buttons, such as the two radio buttons used to specify direction in the Find dialog of the Text Editor application. Only one button in a radio list can be selected at a single time.

Inheritance Hierarchy

- *Control*
  - RadioList

Syntax

```
'Declaration
Public Class RadioList _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the radio list.
Items	A list of items in the radio list.

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The index of the selected item.
SelectedItem	The item to select.
State	The state of a check box. Values include: 1=checked,2=unchecked, 3=undecided
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item from the radiolist.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

RadioListToolItem Class

Description
RadioListToolItem is the class for a radio button element in a toolbar control.

Inheritance Hierarchy

- *ToolItem*
  - RadioListToolItem

Syntax
```
'Declaration
Public Class RadioListToolItem _
Inherits ToolItem
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
State	The state of a check box. Values include: 1=checked,2=unchecked, 3=undecided
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

RegistryCategory Enumeration

Description

Specifies the root key for the registry. For additional information, see *Predefined Keys*.

Inheritance Hierarchy

- *SystemFunctions*
  - `RegistryCategory`

Syntax

```
Public Enum RegistryCategory
```

Members

Name	Description
HKEY_CLASSES_ROOT	Registry entries subordinate to this key define types (or classes) of documents and the properties associated with those types.
HKEY_CURRENT_USER	Registry entries subordinate to this key define the preferences of the current user.
HKEY_LOCAL_MACHINE	Registry entries subordinate to this key define the physical state of the computer.

Name	Description
HKEY_USERS	Registry entries subordinate to this key define the default user configuration for new users on the local computer and the user configuration for the current user.

RegistryView Enumeration

Description

Defines whether to use the 32bit or the 64bit registry.

Inheritance Hierarchy

- *SystemFunctions*
  - RegistryView

Syntax

```
Public Enum RegistryView
```

Members

Name	Description
RegistryView_32Bit	Use the 32bit registry.
RegistryView_64Bit	Use the 64bit registry.

Scale Class

Description

The class for scales.

Inheritance Hierarchy

- *Control*
  - Scale

Syntax

```
'Declaration
Public Class Scale _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)

Name	Description
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	The actual position of the scale caret.
Range	The range of the scale.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetPosition	Sets the position of the scale.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetToMax	Sets the scale to its maximum position.
SetToMin	Sets the scale to its minimum position.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ScrollBar Class

Description

The class for scroll bars, including the scroll bars that are parts of controls, such as the scroll bar on a list box.

Inheritance Hierarchy

- *Control*
  - ScrollBar
    - *HorizontalScrollBar*
    - *VerticalScrollBar*

Syntax

```
'Declaration
Public Class ScrollBar _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	The internal page size of the scroll bar.
Position	The actual position of the scroll bar caret.
Range	The range of the scroll bar.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this

Name	Description
	property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Scrolls the scroll bar.
ScrollToMax	Scrolls the scroll bar to its maximum position.
ScrollToMin	Scrolls the scroll bar to its minimum position.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SeparatorItem Class

Description
The class for menu separators.

Inheritance Hierarchy

- *Item*
  - SeparatorItem

Syntax

```
'Declaration
Public Class SeparatorItem _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

TabControl Class

Description
The class for tabbed, multi-page dialogs and for button bars.

Inheritance Hierarchy

- *Control*
  - TabControl
    - *SWTTabControl*

Syntax
```
'Declaration
Public Class TabControl _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the tab control.
Items	A list of items in the tab control.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The index of the selected item.
SelectedItem	The selected item.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Sets the current page to the specified page.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Table Class

Description
The class for tables.

Inheritance Hierarchy

- *Control*

- Table
  - *SWTTable*

Syntax

```
'Declaration
Public Class Table _
Inherits Control
```

Properties

Name	Description
AllowsCheck	Whether the control can display a checkmark.
AllowsMultiSelect	Whether the table supports selecting multiple items.
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
ColumnCount	The number of columns in the table.
ColumnItems	A list of all items in the column.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
RowCount	The number of rows in the table.
RowItems	A list of all table rows in the table.
SelectedIndices	The indices of the selected item(s).
SelectedItems	The selected item(s).
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box in the defined row of a table.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Double-clicks an item.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
ExtendSelectRow	Selects a range of rows.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
FocusRow	Focuses on a row in the table.
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)

Name	Description
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MultiSelectRow	Adds a row from the table to the set of selected rows.
MultiUnselectRow	Removes a row from the set of selected rows.
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SelectRow	Selects an row in the table.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box in the defined row of a table.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

TableColumn Class

Description
The class for columns in a table.

Inheritance Hierarchy

- *Item*
  - TableColumn
    - *SWTTableColumn*

Syntax

```
'Declaration
Public Class TableColumn _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
Width	The width of the column.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Resize	Resizes a column.
Select	Selects a column.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

TableRow Class

Description
The class for rows in a table.

Inheritance Hierarchy

- *Item*
  - TableRow
    - *SWTTableRow*

Syntax
```
'Declaration
Public Class TableRow _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Checked	Whether the check box in the row is checked.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the row.
Items	A list of items in the row.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

TestObject Class

Description
The base class of all UI objects.

Syntax
```
'Declaration
Public Class TestObject
```

Properties

Name	Description
Text	The text of the control.

Name	Description
Value	The value of the control, e.g.: text in a text control.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately.
Exists	Checks if an object exists in the application under test.
Find	Finds an object specified by an XPath locator.
FindAll	Finds all objects specified by an XPath locator.
GenerateLocator	Returns a locator for this object.
GetChildren	Returns the child objects of this object.
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject.
GetParent	Looks up the parent of this object in the test application.
GetProperty	Returns the value of the specified property.
GetPropertyList	Returns a list of property names that can be retrieved for the given object.
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object.
HighlightObject	Highlights this object.
ImageClick	Clicks on specified image asset.
ImageClickFile	Clicks on the specified image.
ImageExists	Returns whether the specified image asset exists.
ImageExistsFile	Returns whether the specified image exists.
ImageRectangle	Returns the object-relative rectangle of the specified image asset.
ImageRectangleFile	Returns the object-relative rectangle of the specified image.
Invoke	Dynamically invokes a method on the test object.
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject.

Name	Description
SetProperty	Sets the value of the specified property.
TextCapture	Returns the text in this object's visible area.
TextClick	Clicks in the center of the specified text.
TextExists	Returns whether the specified text exists.
TextRectangle	Returns the object-relative rectangle of the specified text.
Verify	Executes a verification on the given asset in the context of this UI object.
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached.
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached.
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts.
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached.

TextField Class

Description
The class for single-line and multi-line fields whose text can be modified by the user. The class also supports the Windows RichEdit control.

Inheritance Hierarchy

- *Control*
  - TextField
    - *StyledText*

Syntax

```
'Declaration
Public Class TextField _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
IsPassword	Whether the control is a password text field.
MultiLine	Whether the control is multiline.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	The position of the caret within the text field.
SelectedRange	The selected range within the text field.
SelectedText	The selected text within the text field. An empty string if no text is selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearText	Removes all text from the text field.
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetMultiText	Returns the specified lines of text in the multi-line text field.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetMultiText	Substitutes all or part of the lines in the multi-line text field.
SetPosition	Sets the insertion point in the text field.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the specified range of the single-line or multi-line text field.
SetText	Substitutes new text for all or part of the text in the text field.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ToggleButton Class

Description
The class for buttons that have a state that can be changed by selecting the button.

Inheritance Hierarchy
- *Control*
  - ToggleButton

Syntax
```
'Declaration
Public Class ToggleButton _
Inherits Control
```

Properties
Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)

Name	Description
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
State	The state of a toggle button. Values include: 1=checked, 2=unchecked, 3=undecided.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ToolBar Class

Description
The class for tool bar controls. A tool bar is a container for a set of buttons and other standard controls.

Inheritance Hierarchy

- *Control*
  - ToolBar

Syntax
```
'Declaration
Public Class ToolBar _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ToolItem Class

Description
The class for a tool item in a toolbar control.

Inheritance Hierarchy

- *Item*
  - ToolItem
    - *CheckBoxToolItem*
    - *DropDownToolItem*
    - *PushToolItem*
    - *RadioListToolItem*

Syntax

```
'Declaration
Public Class ToolItem _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Tree Class

Description
The class for items organized into graphical hierarchies.

Inheritance Hierarchy

- *Control*
  - Tree
    - *SWTTree*

Syntax

```
'Declaration
Public Class Tree _
Inherits Control
```

Properties

Name	Description
AllowsCheck	Whether the control can display a checkmark.
AllowsMultiSelect	Whether the control supports selecting multiple items.
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the tree (including all children).
ItemPaths	A list of items in the tree (including all children).
Items	A list of items in the tree (including all children).
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The index of the first selected item.
SelectedIndices	The indices of the selected item(s).
SelectedItem	The name of the first selected item.

Name	Description
SelectedItems	The names of the selected item(s).
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
VisibleItemPaths	A list of visible items in the tree (including all children).
VisibleItems	A list of visible items in the tree (including all children).

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box.
Click	Clicks on the object. (Inherited from *IClickable*)
Collapse	Collapses an item in a treeview control.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Double-clicks an item.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Expand	Expands an object in a treeview control.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetItemPaths	Returns all items of the tree as a list of itempaths
GetItemRect	Returns the size and position of an item relative to the treeview control.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsExpandable	Checks if the given item can be expanded.
IsExpanded	Checks if the given item is expanded.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item from the tree.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

VerticalScrollBar Class

Description
The class for vertical scroll bars.

Inheritance Hierarchy

- *ScrollBar*
  - VerticalScrollBar

Syntax

```
'Declaration
Public Class VerticalScrollBar _
Inherits ScrollBar
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	The internal page size of the scroll bar. (Inherited from *ScrollBar*)

Name	Description
Position	The actual position of the scroll bar caret. (Inherited from *ScrollBar*)
Range	The range of the scroll bar. (Inherited from *ScrollBar*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PageDown	Scrolls the scroll bar page-wise down.
PageUp	Scrolls the scroll bar page-wise up.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Scrolls the scroll bar. (Inherited from *ScrollBar*)
ScrollDown	Moves the scroll bar down.
ScrollToMax	Scrolls the scroll bar to its maximum position. (Inherited from *ScrollBar*)
ScrollToMin	Scrolls the scroll bar to its minimum position. (Inherited from *ScrollBar*)
ScrollUp	Moves the scroll bar up.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Window Class

Description
The class for windows that typically can move.

Inheritance Hierarchy

- *BaseGuiTestObject*
  - Window
    - *BrowserApplication*
    - *Dialog*
    - *FlexStandalonePlayer*
    - *FormsWindow*
    - *Shell*

Syntax
```
'Declaration
Public Class Window _
Inherits BaseGuiTestObject _
Implements IMoveable
```

Properties

Name	Description
Application	The name of the Application that this Window belongs to.
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout

Name	Description
	parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Core Class Reference

When you configure an application, Silk Test Workbench automatically provides built-in support for testing standard agent and test object controls.

ActiveData Class

Description

`ActiveData` is the class that provides functionality for using an `ActiveData` object in a script. This class is an internal constructor. To call `ActiveData`, call `Workbench.LoadActiveData` instead of using this class.

Inheritance Hierarchy

`ActiveData` does not derive from any class, and no classes derive from it.

Syntax

```
'Declaration
Public Class ActiveData
```

Properties

Name	Description
ColumnCount	The number of columns in the underlying `ActiveData` object.
Count	The number of columns in an `ActiveData` row.

Methods

Name	Description
AddColumn	Adds a column to the underlying ActiveData object.
AddRow	Adds a row to the underlying ActiveData object.
Dispose	Cleans up the resources used by the object.
GetEnumerator	Returns an enumerator for the rows.
Item	Returns an ActiveDataRow by index.
RemoveColumn	Removes a column from the underlying ActiveData object.
RemoveRow	Removes a row from the underlying ActiveData object.
RenameColumn	Renames a column in the ActiveData object.
Reset	Moves the focus to the first row of the ActiveData file.
Save	Saves the ActiveData object.

AddColumn Method (ActiveData)

Class

ActiveData.

Action

Adds a column to the underlying ActiveData object and returns a value that indicates success or failure.

Syntax

```
AddColumn (defaultValue, columnName)
```

Variable	Description
defaultValue	The default value for the new column. Double.
columnName	The name of the column. String.

Examples

To add a new column called "Occupation" to the PhoneBookData asset, type the following:

```
Public Sub Main()

        AddColumnToDataFile( "occupation" )

End Sub

Public Sub AddColumnToDataFile( name As String )
        Dim data As ActiveData = Workbench.LoadActiveData( "PhoneBookData" )
        data.AddColumn( "Occupation", name )
        data.Save()
    End Sub
End Module
```

AddRow Method (ActiveData)

Class

ActiveData.

Action

Adds a row to the underlying ActiveData object and returns a value that indicates success or failure.

Syntax

```
AddRow ()
```

Examples

To add a new row of data for ID number, type the following:

```
Public Sub Main()

        InsertNewNumberToDataFileWithOccupation( 108, "Jay", "Jones",
"5551219", _
    "QA Analyst" )

End Sub

Public Sub InsertNewNumberToDataFileWithOccupation( id As Integer, _
    firstName As String, lastName As String, phoneNumber As String, _
    occupation As String )
        Dim data As ActiveData = Workbench.LoadActiveData( "PhoneBookData" )
        Dim row As ActiveDataRow = data.AddRow()

        row.SetLong( "id", id )
        row.SetString( "fname", firstName )
        row.SetString( "lname", lastName )
        row.SetString( "number", phoneNumber )
        row.SetString( "occupation", occupation )

        data.Save()

End Sub
End Module
```

Dispose Method (ActiveData)

Class

ActiveData.

Action

Cleans up the resources used by the object.

Syntax

```
Dispose ()
```

Notes

The only time that you should use this method is when you are reading and writing to a file from multiple scripts.

Caution: In all cases except previously mentioned, do not use this method. Silk Test Workbench automatically calls this method when it is needed.

Example

```
Dim a As ActiveData
a = Workbench.LoadActiveData("ActiveDataAsset")
' Do your work here
a.Dispose()
```

GetEnumerator Method (ActiveData)

Class

ActiveData.

Action

Return an enumerator for the rows.

Syntax

```
GetEnumerator ()
```

Item Method (ActiveData)

Class

ActiveData.

Action

Returns an `ActiveDataRow` by index.

Syntax

```
Item(index)
```

Variable	Description
index	The index of the row to return. `Integer`.

RemoveRow Method (ActiveData)

Class

ActiveData.

Action

Removes a row from the underlying `ActiveData` object.

Syntax

```
RemoveRow (rowNumber)
```

Variable	Description
rowNumber	The number of the row to remove (1-based). `Integer`.

RemoveColumn Method (ActiveData)

Class

ActiveData.

Action

Removes a column from the underlying `ActiveData` object and returns a value that indicates success or failure.

Syntax

```
RemoveColumn (columnName)
```

Variable	Description
columnName	The number of the row to remove (1-based). `Integer`.

RenameColumn Method (ActiveData)

Class

ActiveData.

Action

Renames a column in the `ActiveData` object and returns a value that indicates whether the column was renamed successfully.

Syntax

```
RenameColumn (columnIndex, newColumnName)
```

Variable	Description
columnIndex	The index of the column to remove (1-based). `Integer`.
newColumnName	The new name for the column. `String`.

Reset Method (ActiveData)

Class

ActiveData.

Action

Moves the focus to the first row of the `ActiveData` file.

Syntax

```
Reset ()
```

Save Method (ActiveData)

Class

ActiveData.

Action

Saves the `ActiveData` object.

Syntax

```
Save ()
```

ColumnCount Property

Class

ActiveData.

Action

Returns the number of columns in the underlying active data object.

Syntax

```
'Declaration
Protected Read Property ColumnCount As Integer
```

Access

Read only.

Count Property

Class

ActiveData.

Action

Returns the number of columns in an `ActiveData` row.

Syntax

```
'Declaration
Protected Read Property Count As Integer
```

Access

Read only.

ActiveDataRow Class

Description

`ActiveDataRow` is the class that provides functionality for using rows in an `ActiveData` object. This class is an internal constructor.

Inheritance

`ActiveDataRow` does not derive from any class, and no classes derive from it.

Syntax

```
'Declaration
Public Class ActiveDataRow
```

Properties

Name	Description
Count	The number of columns in an `ActiveData` row.
ExternalRowNumber	Gets the number of the row of the current object in the set of loaded rows.
RowNumber	Gets the number of the row of the current object in the `ActiveData` file.

Methods

Name	Description
Dispose	Cleans up the resources used by the row.
GetDouble	Gets the double value of a column.
GetLong	Gets the integer value of a column.
GetLongLong	Gets the long value of a column.
GetString	Gets the string of a column.
SetDouble	Sets the double value of a column.
SetLong	Sets the integer value of a column.
SetLongLong	Sets the long value of a column.
SetString	Sets the string value of a column.

Dispose Method (ActiveDataRow)

Class

ActiveDataRow.

Action

Cleans up the resources used by the row.

Syntax

```
Dispose ()
```

Notes

Caution: Do not use this method. Silk Test Workbench automatically calls this method when it is needed.

GetDouble Method (ActiveDataRow)

Class

ActiveDataRow.

Action

Gets the double value of a column.

Syntax

```
GetDouble (columnIndex)
```

or

```
GetDouble (columnName)
```

Variable	Description
columnIndex	The index of the column (1-based). `Integer`.
columnName	The name of the column. `String`.

GetLong Method (ActiveDataRow)

Class

ActiveDataRow.

Action

Gets the integer value of a column.

Syntax

```
GetLong (columnIndex)
```

or

```
GetLong (columnName)
```

Variable	Description
columnIndex	The index of the column (1-based). `Integer`.
columnName	The name of the column. `String`.

GetLongLong Method (ActiveDataRow)

Class

ActiveDataRow.

Action

Gets the long value of a column.

Syntax

```
 GetLongLong (columnIndex)
```

or

```
GetLongLong (columnName)
```

Variable	Description
columnIndex	The index of the column (1-based). `Integer`.
columnName	The name of the column. `String`.

GetString Method (ActiveDataRow)

Class

ActiveDataRow.

Action

Gets the string of a column.

Syntax

```
GetString (columnIndex)
```

or

```
GetString (columnName)
```

Variable	Description
columnIndex	The index of the column (1-based). `Integer`.
columnName	The name of the column. `String`.

SetDouble Method (ActiveDataRow)

Class

ActiveDataRow.

Action

Sets the double value of a column.

Syntax

```
SetDouble (columnIndex, value)
```

or

```
SetDouble (columnName, value)
```

Variable	Description
columnIndex	The index of the column (1-based). `Integer`.
columnName	The name of the column. `String`.
value	The new value. `Integer` or `Double`.

SetLong Method (ActiveDataRow)

Class

ActiveDataRow.

Action

Sets the integer value of a column.

Syntax

```
SetLong (columnIndex, value)
```

or

```
SetLong (columnName, value)
```

Variable	Description
columnIndex	The index of the column (1-based). `Integer`.
columnName	The name of the column. `String`.
value	The new value. `Integer`.

SetLongLong Method (ActiveDataRow)

Class

ActiveDataRow.

Action

Sets the long value of a column.

Syntax

```
SetLongLong (columnIndex, value)
```

or

```
SetLongLong (columnName, value)
```

Variable	Description
columnIndex	The index of the column (1-based). `Integer`.
columnName	The name of the column. `String`.
value	The new value. `Long`.

SetString Method (ActiveDataRow)

Class

ActiveDataRow.

Action

Sets the string value of a column.

Syntax

```
SetString (columnIndex, value)
```

or

```
SetString (columnName, value )
```

Variable	Description
columnIndex	The index of the column (1-based). `Integer`.
columnName	The name of the column. `String`.
value	The new value. `String`.

Count Property

Class

ActiveData.

Action

Returns the number of columns in an `ActiveData` row.

Syntax

```
'Declaration
Protected Read Property Count As Integer
```

Access

Read only.

ExternalRowNumber Property

Class

ActiveDataRow.

Action

Gets the row number the current object is in the set of loaded rows.

Syntax

```
'Declaration
Protected Read Property ExternalRowNumber As Integer
```

Access

Read only.

RowNumber Property

Class

ActiveDataRow.

Action

Gets the row number the current object is in the actual file.

Syntax

```
'Declaration
Protected Read Property RowNumber As Integer
```

Access

Read only.

Agent Class

Description

`Agent` is the class for the Open Agent, the component of Silk Test Workbench that interacts with the graphical user interface.

`Agent` has a predefined identifier, Agent. To call a method of the `Agent`, use this syntax:

```
Agent.method()
```

Inheritance Hierarchy

`Agent` does not derive from any class, and no classes derive from it.

Syntax

```
'Declaration
Public Class Agent
```

Properties

Name	Description
Desktop	Gets the root GUI object.

Methods

Name	Description
Attach	Attaches to the given application. Enables you to specify an additional command-line pattern and tech domain.
Connect	Connects to an agent on the given machine by using the specified host and port.
Decrypt	Decrypts the given string. Only strings that were encrypted with the *Encrypt* method can be decrypted.
DetachAll	Detaches all currently attached applications.
Encrypt	Encrypts the given string.
ExecuteBaseState	Ensures that the test application is running and ready for testing.
GetOption	Retrieves the value of an agent option.
ResetOptions	Resets all agent options to their default values.

Name	Description
SetOption	Sets an agent option. This option is distributed to all technology domains.
Shutdown	Stops the agent. Should be called only when the test execution is finished.

Attach Method

Class

Agent.

Action

Attaches to the given application. Enables you to specify an additional command-line pattern and tech domain.

Syntax

```
Attach (executablePattern, commandLinePattern, techDomains)
```

Variable	Description
executablePattern	The executable name of the application. The pattern may include the wildcards '?' and '*' which match one or none to many characters respectively. For example `myApplication.exe`, `myApplica?ion.exe`, and `myApp*.exe`. `String`.
commandLinePattern	*Optional:* This pattern is matched against the command line arguments of the application. This is useful when multiple instances of an application are running, but only some of them should be tested. For Java applications, the command line pattern could include the name of a characteristic jar or the main class, for example `*org.MyMainClass`. You can use `""` to specify no command line pattern when you want to load specific tech domains. `String`.
techDomains	*Optional:* The tech domains to load in the application. Typically, this parameter is only used by technical support. As a best practice, do not specify a tech domain. If you do not specify a tech domain, all available tech domains are loaded. `TechDomain`.

Examples

The following example attaches to the Notepad application.

```
Agent.Attach("notepad.exe")
```

The following example shows how to use the command-line pattern to attach to a specific instance of a process when multiple instances of the application are running.

```
Agent.Attach("javaw.exe", “*org.MyMainClass”)
```

The following example shows how to attach to Internet Explorer and load the Win32 tech domain:

```
Agent.Attach("iexplore.exe", "", {"WIN32"})
```

Connect Method

Class

Agent.

Action

Connects to an agent on the given machine by using the specified host and port.

If no agent is running or the agent cannot be reached on that machine, an exception is thrown.

If the specified host is the localhost, the agent is started.

After the connection to this agent is established for the first time, all agent options are reset to their default values by calling `Agent.ResetOptions()`.

Syntax

```
Connect (host, [port])
```

Variable	Description
host	The host of the Open Agent. `String`.
port	*Optional:* The port of the Open Agent. If a port is not specified, the default port is used. This port is the information service port with the default value of 22901. For example, to connect the agent to a different information service port, type `Agent.Connect("myRemoteMachine", portnumber)` If you are connected to a remote agent, you can connect to your local agent by typing `Agent.Connect("localhost")`. `Integer`.

Decrypt Method

Class

Agent.

Action

Decrypts the given string. Only strings can be decrypted that were encrypted with the `Encrypt` method.

Syntax

```
Decrypt(toDecrypt)
```

Variable	Description
toDecrypt	The string to decrypt. `String`.

DetachAll Method

Class

Agent.

Action

Detaches all currently attached applications.

Syntax

```
DetachAll( )
```

Encrypt Method

Class

Agent.

Action

Encrypts the given string.

Syntax

```
Encrypt (toEncrypt)
```

Variable	Description
toEncrypt	The string to encrypt. `String`.

ExecuteBaseState Method

Class

Agent.

Action

Ensures that the test application is running and ready for testing.

Syntax

```
TestObject ExecuteBaseState(baseState)
```

Variable	Description
baseState	Contains all information required for executing the base state. `IBaseState`.

GetOption Method

Class

Agent.

Action

Retrieves the value of an agent option.

Syntax

```
GetOption(name)
```

Variable	Description
name	The name of the option. A list of available options are defined in the `Options` class. `String`.

Example

For example, you might type:

```
desktop.getOption(Options.ObjectResolveTimeout)
```

ResetOptions Method

Class

Agent.

Action

Resets all agent options to their default values.

Syntax

```
ResetOptions()
```

SetOption Method

Class

Agent.

Action

Sets an agent option. This option is distributed to all Tech Domains.

Syntax

```
SetOption(name, value)
```

Variable	Description
name	The name of the option. A list of available options are defined in the `Options` class. `String`.
value	The value of the option. `Object`.

Example

For example, you might type:

```
Agent.setOption(Options.ObjectResolveTimeout, true)
```

Shutdown Method (Agent)

Class

Agent.

Action

Call this method to ensure that the Open Agent stops after a test run, for example after a nightly test execution. You should only call this method at the end of a test execution, and not before all actions of the test are executed. If any actions would follow the `Shutdown`, the agent would be restarted. Alternatively, the following code sample shows how you can stop the Open Agent from the command line:

```
openAgent.exe -shutDown
```

Syntax

VB

```
Shutdown()
```

Example: Using the shutdown method

```
// VB code
<AssemblyCleanup()>
Public Shared Sub ShutdownAgentAfterTestRun()
  Agent.Shutdown()
End Sub
```

Desktop Property

Class

Agent.

Action

Gets the root GUI object.

Syntax

```
'Declaration
Protected Read Property Desktop As Desktop
```

Access

Read only.

BaseState Class

Description

The `BaseState` class ensures that the application under test is running and ready for testing. Additionally the class is used to bring the application to the front.

If you try to attach the agent to the application and the test object, which is specified by the `BaseState.Locator`, is found, the base state brings the window containing the test object to the front and returns the found test object immediately. If the test object is not found the application is started with the command line as specified in `baseStateInfo` and the agent attaches to the application. The agent waits until the test object is found. You can specify the timeout for the wait with the option

`Options`.ObjectResolveTimeout. The default value for the timeout is 30 seconds. To change the timeout see `Desktop.SetOption(String, Object)`. You can also change the timeout in the UI by editing the **Application ready timeout** under **Playback** > **Timing** in the **Options** dialog box. If the test object is found within the given timeout the window that contains the test object is brought to the front and the found test object is returned. If no test object is found after the timeout an exception is thrown.

Inheritance Hierarchy

`BaseState` does not derive from any class, and no classes derive from it.

Syntax

```
'Declaration
Public Class BaseState _
Implements IBaseState
```

Properties

Name	Description
CommandLineArguments	Gets or sets the command-line arguments.
CommandLinePattern	Gets or sets the command-line pattern.
ExecutablePattern	Gets or sets the executable pattern.
Executable	Gets the name of the executable.
Locator	Returns the locator of the executable.
TechDomains	Gets or sets the list of tech domains to be used for testing.
WorkingDirectory	Gets or sets the working directory.

CommandLineArguments Property

Class

BaseState.

Action

Gets or sets the command-line arguments.

Syntax

```
'Declaration
Public ReadWrite Property CommandLineArguments As String
```

Access

Read and write.

CommandLinePattern Property

Class

BaseState.

Action

Gets or sets the command-line pattern.

Syntax

```
'Declaration
Public ReadWrite Property CommandLinePattern As String
```

Access

Read and write.

ExecutablePattern Property

Class

BaseState.

Action

Gets or sets the executable pattern.

Syntax

```
'Declaration
Public ReadWrite property ExecutablePattern As String
```

Access

Read and write.

Executable Property

Class

BaseState.

Action

Returns the name of the executable.

Syntax

```
'Declaration
Public Read Property CommandLineArguments As String
```

Access

Read only.

Locator Property

Class

BaseState.

Action

Gets the locator of the executable.

Syntax

```
'Declaration
Public Read Property Locator As String
```

Access

Read only.

TechDomains Property

Class

BaseState.

Action

Gets the technology domains that are necessary for an executable.

Syntax

```
'Declaration
Public Read Property TechDomains As List of TechDomain
```

Access

Read only.

WorkingDirectory Property

Class

BaseState.

Action

Gets or sets the working directory. Can contain Windows environment variables.

Syntax

```
'Declaration
Public ReadWrite Property WorkingDirectory As String
```

Access

Read and write.

BrowserBaseState Class

Description

Ensures that the browser that is specified by the `Executable` property is running and ready for testing. The base state additionally navigates to the URL that is specified by the `Url` property and brings the browser to the front. The browser base state is executed as follows:

- Try to attach the agent to the browser.
  - If the test object, which is specified by the `Locator` property, is found, the base state navigates to the given URL, brings the browser to the front, and returns the found test object.

    - If the test object is not found the browser is started with the command line as specified in `browserBaseStateInfo` and the agent attaches to the browser.
- The agent waits until the test object is found. The timeout can be specified with the option `Options`.ObjectResolveTimeout. To change the timeout see the `SetOption` method of the `Desktop` class.
    - If the test object is found within the given timeout the base state navigates to the given URL, brings the browser to the front and returns the found test object.
    - If no test object is found after the timeout an Exception is thrown.

Inheritance Hierarchy

`BrowserBaseState` does not derive from any class, and no classes derive from it.

Syntax

```
'Declaration
Public Class BrowserBaseState _
Implements IBaseState
```

Properties

Name	Description
CommandLineArguments	Gets or sets the command-line arguments. (Inherited from *BaseState*)
CommandLinePattern	Gets or sets the command-line pattern. (Inherited from *BaseState*)
ExecutablePattern	Gets or sets the executable pattern. (Inherited from *BaseState*)
Executable	Gets the name of the executable. (Inherited from *BaseState*)
Locator	Returns the locator of the executable. (Inherited from *BaseState*)
TechDomains	Gets or sets the list of tech domains to be used for testing. (Inherited from *BaseState*)
Url	Gets the Url to navigate to.
WorkingDirectory	Gets or sets the working directory. (Inherited from *BaseState*)

Url Property

Class

BrowserBaseState.

Action

Gets the URL to navigate to.

Syntax

```
'Declaration
Public Read Property Url As String
```

Access

Read only.

ConsoleWindow Class

Description

`ConsoleWindow` is the class for consoles.

Inheritance Hierarchy

`ConsoleWindow` does not derive from any class, and no classes derive from it.

Syntax

```
'Declaration
Public Class ConsoleWindow
```

Methods

Name	Description
`Close`	Closes the console.
`CloseAll`	Closes all open consoles or all consoles with a specific caption.
`GetContents`	Returns the contents of the console.
`Open`	Opens a new console.
`TypeKeys`	Sends a set of keystrokes to the console.

Close Method (ConsoleWindow)

Class

`ConsoleWindow`.

Action

Closes the console.

Syntax

```
consoleWindow.Close ([sItemIdentifier])
```

Variable	Description
sItemIdentifier	The index of the console window or the caption of the window. If this parameter is left blank, Silk Test Workbench closes the first found console window. `STRING`.

Example 1

The following sample shows how you can use the index to define that you want to close the second console window:

```
consoleWindow.Close(2)
```

Example 2

The following sample shows how you can use the full identifier of a console window to close the console window:

```
consoleWindow.Close("C:\\Windows\\system32\\cmd.exe[1]")
```

CloseAll Method (ConsoleWindow)

Class

ConsoleWindow.

Action

Closes all open consoles or all consoles with a specific caption.

Syntax

VB

```
consoleWindow.CloseAll ([sCaption])
```

Variable	Description
sCaption	Set this parameter to close only the consoles with this caption. `STRING`

Example

The following sample shows how you close all console windows with the caption `MyCommandWindow`:

```
// VB
consoleWindow.CloseAll("MyCommandWindow")
```

GetContents Method (ConsoleWindow)

Class

ConsoleWindow.

Action

Returns the contents of the console.

Syntax

```
lsContents = consoleWindow.GetContents ([sItemIdentifier])
```

Variable	Description
lsContents	The console contents. `LIST OF STRING`.
sItemIdentifier	The index of the console window or the caption of the window. If this parameter is left blank, Silk Test Workbench returns the contents of the first found console window. `STRING`.

Notes

- Each string in the returned list of strings corresponds to one row in the console.

Example 1

The following sample shows how you can use the index to define that you want to get the contents of the second console window:

```
lsContents = consoleWindow.GetContents(2)
```

Example 2

The following sample shows how you can use the full identifier of a console window to get the contents of the console window:

```
lsContents = consoleWindow.GetContents("C:\\Windows\\system32\
\cmd.exe[1]")
```

Open Method (ConsoleWindow)

Class

`ConsoleWindow`.

Action

Opens a new console.

Availability

This functionality is supported only if you are using the Open Agent.

Syntax

VB

```
ConsoleWindow.Open ([caption])
```

Variable	Description
caption	The caption for the new console window. `String`.

Example

The following sample shows how you can open a new console with the caption "My Console":

```
// VB
ConsoleWindow.Open("My Console")
```

TypeKeys Method (ConsoleWindow)

Class

ConsoleWindow.

Action

Sends a set of keystrokes to the console.

Syntax

```
ConsoleWindow.TypeKeys (sEvents [, nDelay, ensureFocus, sItemIdentifier])
```

Variable	Description
sEvents	The keystrokes to type. `STRING`.
nDelay	*Optional:* The delay between keystrokes (in seconds and fractions of seconds). `NUMBER`.
ensureFocus	*Optional:* Whether the agent brings the application to the foreground and then sets the keyboard focus to the control on which the method is executed. This is executed before the first key is typed, which means it is not rechecked for each keystroke when you specify multiple keys to be typed. By default, this variable is set to TRUE. `BOOLEAN`.
sItemIdentifier	The index of the console window or the caption of the window. If this parameter is left blank, Silk Test Workbench sends the keystrokes to the first found console window. `STRING`.

Notes

- If no console is open, `TypeKeys` opens a console.
- On Windows, `TypeKeys` first gives the console input focus before typing into it.
- If you specify *nDelay*, that value overrides the value set with the `OPT_KEYBOARD_DELAY` option, which you set with `SetOption`.
- If keystrokes are not recognized, you can try to use a higher value for the delay.
- A key can be followed by a counter, which defines how often the keys should be typed. The counter can be combined with the delay.
- During the delay time, the screen is locked and interaction with the mouse and keyboard is no longer possible.
- When recording with the DOM extension `TypeKeys` ("<Tab>") will not be captured. Since the script refers to object to type in directly, it is not necessary to record this manual Tab. You can manually enter a `TypeKeys` ("<Tab>") into your script if you want to; it just will not be recorded.
- When specifying groups of keys, such as **Ctrl-Shift**, the agent supports plus and minus signs.

Example

The following code sample executes the `dir` command in the console window to list the files and folders in the directory:

```
// VB
ConsoleWindow.TypeKeys ("dir<Enter>")
```

DllCall Class

Description

The `DllCall` class allows NTF users to call exposed DLL functions of any DLL either on the Agent or in the AUT.

Inheritance Hierarchy

`DllCall` does not derive from any class, and no classes derive from it.

Syntax

```
'Declaration
Public Class Agent
```

Methods

Name	Description
UnloadAllDllsFromAgent	Unloads all DLLs from the agent.
UnloadDllFromAgent	Unloads the specified DLL from the agent.

UnloadAllDllsFromAgent Method

Class

DllCall.

Action

Unloads all DLLs from the agent.

Syntax

```
Agent.UnloadAllDllsFromAgent ()
```

UnloadDllFromAgent Method

Class

DllCall.

Action

Unloads the specified DLL from the Agent.

Syntax

```
bSuccessful = Agent.UnloadDll (sDllName)
```

Variable	Description
bSuccessful	True if the DLL was successfully unloaded, false if not. BOOLEAN.
sDllName	The name of the DLL file. STRING.

Notes

You can use `UnloadDllFromAgent` only to unload normal DLLs, not mixed assemblies. To unload mixed assemblies, use `UnloadAllDllsFromAgent`.

IBaseState Interface

Description

Interface for base states. A base state makes an application ready for testing. It launches it if necessary, wait for a certain locator to be found and brings the application to the front.

Syntax

```
'Declaration
Public Interface IBaseState
```

Methods

Name	Description
Execute	Executes the base state on the machine as specified by the desktop.

Execute Method

Class

BaseState.

Action

Executes the base state on the machine as specified by the desktop.

Syntax

```
TestObject = BaseState.Execute()
```

Variable	Description
TestObject	The object for which the locator should wait.

Example: Specify the browser and the URL of a web application in the base state

You can use the following code to specify Internet Explorer as the browser on which a web application should be tested and *www.borland.com* as the URL of the web application:

```
Dim baseState = New
BrowserBaseState(BrowserType.InternetExplorer,
"www.borland.com")
baseState.Execute()
```

To additionally store the base state in a variable:

```
Dim baseState = New
BrowserBaseState(BrowserType.InternetExplorer,
"www.borland.com")
Dim browserApplication As BrowserApplication =
baseState.Execute()
```

To specify a mobile browser on a mobile device, for example Safari on an iOS device, you can use the following code:

```
Dim baseState = New BrowserBaseState(BrowserType.Safari,
"www.borland.com")
```

```
baseState.MobileDeviceName = "My iPhone"
baseState.Execute()
```

Example: Specify the path to the executable and a locator for the main window of the AUT

You can use the following code to specify `C:/windows/system32/notepad.exe` as the path to the executable of the AUT and *//Window[@caption='*Notepad*']* as the locator for the main window of the AUT:

```
Dim baseState = New BaseState("C:/windows/system32/
notepad.exe", "//Window[@caption='*Notepad*']")
baseState.Execute()
```

To additionally store the base state in a variable:

```
Dim baseState = New BaseState("C:/windows/system32/
notepad.exe", "//Window[@caption='*Notepad*']")
Dim mainWindow As Window = baseState.Execute()
```

Timer Class

Description

The `Timer` class enables you to accurately measure elapsed time.

Among other usages, the methods and properties in the `Timer` class are used for the timing of test executions that are triggered from Silk Performer. For additional information on integrating Silk Test Workbench with Silk Performer, refer to the *Silk Performer Help*.

Inheritance Hierarchy

`BaseState` does not derive from any class, and no classes derive from it.

Syntax

```
'Declaration
Public Class Timer
```

Methods

Name	Description
Pause	Pauses the time measurement.
Resume	Resumes the time measurement.
Start	Starts or resumes the time measurement.
StartNew	Initializes a new `Timer` instance, sets the elapsed time to zero, and starts measuring elapsed time.
Stop	Stops the time measurement.

Properties

Name	Description
Elapsed	Gets the elapsed time span of the timer.

Name	Description
ElapsedMilliseconds	Gets the elapsed time span of the timer in milliseconds.
IsRunning	Gets whether the timer is currently running or not.
Name	Gets or sets the name of the timer which is used for interfacing with Silk Performer.

Pause Method (Timer)

Class

Timer.

Action

Pauses the time measurement.

Syntax

```
timer.Pause ()
```

Resume Method (Timer)

Class

Timer.

Action

Resumes the time measurement.

Syntax

```
timer.Resume ()
```

Start Method (Timer)

Class

Timer.

Action

If the timer is currently stopped, starts the time measurement. If the timer is currently paused, resumes the time measurement.

Syntax

```
timer.Start ()
```

StartNew Method (Timer)

Class

Timer.

Action

Initializes a new `Timer` instance, sets the elapsed time to zero, and starts measuring elapsed time. The name is provided for interfacing with Silk Performer.

Syntax

```
timer=StartNew ([timerName])
```

Variable	Description
timer	The new `Timer` instance that has just begun measuring elapsed time. `Timer`.
timerName	*Optional:* The name of the new timer. The name is provided for interfacing with Silk Performer. `String`.

Stop Method (Timer)

Class

Timer.

Action

Stops the time measurement.

Syntax

```
timer.Stop()
```

Elapsed Property (Timer)

Class

Timer.

Action

Gets the elapsed time of the timer.

Syntax

```
'Declaration
Public Read Property Elapsed As TimeSpan
```

Access

Read.

ElapsedMilliseconds Property (Timer)

Class

Timer.

Action

Gets the elapsed time of the timer in milliseconds.

Syntax

```
'Declaration
Public Read Property ElapsedMilliseconds As Long
```

Access

Read.

IsRunning Property (Timer)

Class

Timer.

Action

Gets whether the timer is currently running or not. Returns true if the timer is running.

Syntax

```
'Declaration
Public Read Property IsRunning As Boolean
```

Access

Read.

Name Property (Timer)

Class

Timer.

Action

Gets or sets the name of the timer which is used for interfacing with Silk Performer.

Syntax

```
'Declaration
Public ReadWrite Property Name As String
```

Access

Read and write.

Workbench Class

Description

`Workbench` is the class that provides functionality for adding verifications and running child scripts.

Inheritance hierarchy

`Workbench` does not derive from any class, and no classes derive from it.

Syntax

```
'Declaration
Public Class Workbench
```

To call a method of the `Workbench`, use this syntax:

```
Workbench.method()
```

Methods

Name	Description
LoadActiveData	Load an `ActiveData` object to use it in a script.
ResultComment Method	Adds a comment as a step to the **Details** tab of the **Result** window.
RunScript	Runs a child script, which is identified by its name, and passes the parameters of the child script.
Verify	Verifies that the expected value equals the actual value and enables you to add a comment.
VerifyAsset	Executes a verification asset.

LoadActiveData Method

Class

Workbench.

Action

Load an `ActiveData` asset for use by a script.

Syntax

```
Workbench.LoadActiveData("activeDataName", [sheetName, startRow, endRow,
randomCount, readOnly])
```

Variable	Description
activeDataName	The name of the `ActiveData` asset that you want the script to use. `STRING`.  **Note:** If your environment uses multiple projects, use a project qualifier to insert an `ActiveData` asset, which is not uniquely named, from the Common project. For details, see the example.
sheetName	*Optional:* The name of the sheet that is used. The sheet specified here overrides the sheet that is specified in the ActiveData asset. By default, the sheet that is specified in the asset is used. `STRING`
startRow	*Optional:* The index of the row in the `ActiveData` asset to load first. `INTEGER`.
endRow	*Optional:* The index of the row in the `ActiveData` asset to load last. `INTEGER`.
randomCount	*Optional:* Determines if and how records from the `ActiveData` file associated with the asset are used in the `ActiveData` test. A value of 0 or False turns off random mode. The value of -1 or True uses all rows from the `StartRow` value through `EndRow` value in a random order in the `ActiveData` test. A value of 1

Variable	Description
	through N, where N is the number of rows in the Test Data asset, uses that number of random rows from the range from `StartRow` to `EndRow`. For additional information about using different `StartRow` and `EndRow` values with the acceptable values for the `RandomCount` parameter, see *Determining Data Use for ActiveData*. `INTEGER`.
readOnly	*Optional:* Determines whether to open the `ActiveData` asset as read/write or read only. Set to False to open the `ActiveData` asset with read/write permission. Set to True to open the `ActiveData` asset as read-only. If set to read-only, any operation that attempts to write to the `ActiveData` asset in the `ActiveData` testing portion of a script generates a run-time error. `ActiveData` asset are opened for read/write permission by default. `BOOLEAN`.

Examples

To include random rows from an `ActiveData` in a script, you might type:

```
Public Sub Main()
  AddRandomToPhoneBook( 3 )    'Reads 3 rows in a random order
End Sub

Public Sub AddRandomToPhoneBook( howMany As Integer )
  Dim data As ActiveData = Workbench.LoadActiveData( "PhoneBookData", 1, -1,
howMany )
  Dim row As ActiveDataRow

  With _desktop.BrowserWindow("/BrowserApplication//BrowserWindow")
    For Each row In data
      Dim FirstName As String = row.GetString("fname")
      Dim LastName As String = row.GetString("lname")
      Dim PhoneNumber As String = row.GetString("number")

      .DomTextField("@id='txtFirstName'").SetText(FirstName)
      .DomTextField("@id='txtLastName'").SetText(LastName)
      .DomTextField("@id='txtPhoneNumber'").SetText(PhoneNumber)

      .DomButton("@id='btnAdd'").Select()
    Next
  End With
End Sub
```

To include a portion of data from an `ActiveData` file in a script, you might type:

```
Public Sub Main()
  AddSomeToPhoneBook( 3, 5 )    'Reads rows 3, 4 and 5 only
End Sub

Public Sub AddSomeToPhoneBook( firstToAdd As Integer, lastToAdd As Integer )
  Dim data As ActiveData = Workbench.LoadActiveData( "PhoneBookData",
firstToAdd, lastToAdd )
  Dim row As ActiveDataRow

  With _desktop.BrowserWindow("/BrowserApplication//BrowserWindow")
    For Each row In data
      Dim FirstName As String = row.GetString("fname")
      Dim LastName As String = row.GetString("lname")
      Dim PhoneNumber As String = row.GetString("number")

      .DomTextField("@id='txtFirstName'").SetText(FirstName)
      .DomTextField("@id='txtLastName'").SetText(LastName)
```

```
      .DomTextField("@id='txtPhoneNumber'").SetText(PhoneNumber)

      .DomButton("@id='btnAdd'").Select()
    Next
  End With
End Sub
```

To include all data as read only from an `ActiveData` file in a script, you might type:

```
Public Sub Main()
  AddAllToPhoneBookReadOnly()    'Opens the data file as read only
End Sub

Public Sub AddAllToPhoneBookReadOnly()
  Dim data As ActiveData = Workbench.LoadActiveData( "PhoneBookData", 1, -1,
0, True )
  Dim row As ActiveDataRow

  With _desktop.BrowserWindow("/BrowserApplication//BrowserWindow")
    For Each row In data
      Dim FirstName As String = row.GetString("fname")
      Dim LastName As String = row.GetString("lname")
      Dim PhoneNumber As String = row.GetString("number")

      .DomTextField("@id='txtFirstName'").SetText(FirstName)
      .DomTextField("@id='txtLastName'").SetText(LastName)
      .DomTextField("@id='txtPhoneNumber'").SetText(PhoneNumber)

      .DomButton("@id='btnAdd'").Select()
    Next
  End With
End Sub
```

If your environment uses multiple projects, use a project qualifier to insert an `ActiveData` file, which is not uniquely named, from the Common project. For example, you might type:

```
Public Sub Main()
  AddAllToPhoneBookReadOnly()    'Opens the data file as read only
End Sub

Public Sub AddAllToPhoneBookReadOnly()
  Dim data As ActiveData = Workbench.LoadActiveData( "Common.PhoneBookData",
1, -1, 0, True )
  Dim row As ActiveDataRow

  With _desktop.BrowserWindow("/BrowserApplication//BrowserWindow")
    For Each row In data
      Dim FirstName As String = row.GetString("fname")
      Dim LastName As String = row.GetString("lname")
      Dim PhoneNumber As String = row.GetString("number")

      .DomTextField("@id='txtFirstName'").SetText(FirstName)
      .DomTextField("@id='txtLastName'").SetText(LastName)
      .DomTextField("@id='txtPhoneNumber'").SetText(PhoneNumber)

      .DomButton("@id='btnAdd'").Select()
    Next
  End With
End Sub
```

Note: Assets in any project can access assets in the Common project. However, the reverse is not true. To use an `ActiveData` file, from the project in which you are currently working, you do not need to specify a project qualifier. Specify the "Common" project qualifier to use an `ActiveData` file from the Common project in another project.

RunScript Method

Class

Workbench.

Action

Runs a child script, which is identified by its name, and passes its parameters.

Syntax

```
Workbench.RunScript("scriptname", [parameters])
```

Variable	Description
scriptname	The name of the child script that you want to run. `STRING`. **Note:** Assets in any project can access assets in the Common project. However, the reverse is not true. To use a script from the project in which you are currently working, you do not need to specify a project qualifier. Specify the "Common" project qualifier to use a script from the Common project in another project.
parameters	*Optional:* The parameters that you want to pass to the main script from the child script. `STRING` or `OBJECT`.

Note: Scripts stored in Oracle databases are case sensitive. Any included script from an Oracle database must match the name in the database exactly or Silk Test Workbench fails to load the script.

Examples

If your environment uses multiple projects, use a project qualifier to run a script, which is not uniquely named, from the Common project. For example, you might type:

```
Workbench.RunScript("Common.AddAccount")
```

where *AddAccount* is a script that resides in the Common project as well as the project that you are currently using, and you want to use the script from the Common project rather than your current project.

Verify Method

Class

Workbench.

Action

Verifies that the expected value equals the actual value and enables you to add a comment.

Syntax

This method has many variations:

```
result = Workbench.Verify(condition[, VerifyFlags])
```

or

```
result = Workbench.Verify(condition[, comment, VerifyFlags])
```

or

```
result = Workbench.Verify(expected, actual[, VerifyFlags])
```

or

```
result = Workbench.Verify(expected, actual[, VerifyFlags, comment])
```

Variable	Description
result	Whether the verification was successful. BOOLEAN.
condition	The condition that needs to be met. BOOLEAN.
expected	The value that you expect the script to return. OBJECT.
actual	The actual value that the script returns. OBJECT.
comment	*Optional:* The comment to include. STRING.
VerifyFlags	*Optional:* Whether a screenshot should be captured if the verification fails. This is an enumeration with the following possible values: • None • ScreenShotOnFailure VerifyFlags.

Examples

To compare the expected value with the actual value and add a comment, type:

```
Workbench.Verify(expected As Object, actual As Object, comment As String)
```

For example, `Workbench.Verify("red", "green", "checking colors")` fails with the message `checking colors - Actual: [green]; Expected: [red]`.

To compare the expected value with the actual value, to add a comment, and to add a screenshot to the result file if the verification fails, type:

```
Workbench.Verify(expected As Object, actual As Object, comment As String,
verifyFlags As VerifyFlags)
```

For example, `Workbench.Verify("red", "green", "checking colors", verifyFlags.ScreenShotOnFailure)` fails with the message `checking colors - Actual: [green]; Expected: [red]` and adds a screenshot to the result file.

To verify the value returned by the expected result and add a comment, type:

```
Workbench.Verify(condition As Boolean, comment As String)
```

For example, `Workbench.Verify(True, "Test Passed")` passes. While `Workbench.Verify(False, "Test Failed")` fails.

To verify the value returned by the expected result, to add a comment, and to add a screenshot to the result file if the verification fails, type:

```
Workbench.Verify(condition As Boolean, comment As String,
verifyFlags As VerifyFlags)
```

For example, `Workbench.Verify(True, "Test Passed", verifyFlags.ScreenShotOnFailure)` passes and adds no screenshot. While `Workbench.Verify(False, "Test Failed", ScreenShotOnFailure)` fails and adds a screenshot to the result file.

To compare the actual value with the expected value for IEnumerable objects, such as lists and arrays, type:

```
Workbench.Verify(expectedEnumerable, actualEnumerable)
```

For example:

```
Dim selectedItemsList = listBox.SelectedItems ' we assume that a list with
the items "red" and "blue" is returned
Dim expectedItemsList = New List(Of String)()
expectedItemsList.Add("red")
expectedItemsList.Add("blue")

Workbench.Verify(selectedItemsList, expectedItemsList)  ' verification passes

Dim expectedItemsArray = New String() { "red", "blue" }
Workbench.Verify(selectedItemsList, expectedItemsArray) ' verification passes
```

Note: Two IEnumerable objects are considered equal if both have the same number of elements, and the elements are equal and in the same order.

Note: Mathematical operations with floating point numbers may lead to two numbers that are not completely identical because of their internal representation, although they should be equal from a user point of view. Therefore, floating point numbers (Double, Single) are considered equal if their difference is less than 0.00001. At times, this value may not be correct for the situation. In this case, compare the two values with the `Verify(result As Boolean)` instead.

VerifyAsset Method (Workbench)

Class

Workbench.

Action

Executes a verification asset. Throws an ObjectNotFoundException if the UI object to verify cannot be found.

Syntax

```
result = Verification.VerifyAsset(verificationAsset[, verifyFlags)
```

Variable	Description
result	Whether the verification was successful. BOOLEAN.
verificationAsset	The name of the verification asset to execute. STRING.
VerifyFlags	*Optional:* Whether a screenshot should be captured if the verification fails. This is an enumeration with the following possible values: • None • ScreenShotOnFailure VerifyFlags.

Examples

To execute a verification of the image asset *myImageAsset*, type:

```
Workbench.VerifyAsset("myImageAsset")
```

To execute a verification of the image asset *myImageAsset* and to add a screenshot to the result file if the verification fails, type:

```
Workbench.VerifyAsset("myImageAsset", verifyFlags.ScreenShotOnFailure)
```

ResultComment Method

Class

Workbench.

Action

Adds a comment as a step to the **Details** tab of the **Result** window. Result comments allow you to add information about a test to the results of the test. After playback of the test, the comment appears as a step in the **Details** tab of the **Result** window.

Syntax

```
Workbench.ResultComment(comment)
```

Variable	Description
comment	The comment to add to the results. STRING.

Examples

If you want to point out in the results that the test did not perform a verification of the format of some inserted value, you could add a line like the following to your script:

```
Workbench.ResultComment("Note: This test did not verify that the format of
the inserted value was correct.")
```

Data Types

This section lists the various types of data.

Note: For information on the .NET framework data types, for example `Integer`, `Boolean`, `String`, and so on, refer to the MSDN. For information about the data types that you can use in visual tests, see *Variables in Visual Tests*.

ClickType Enumeration

Description

Specifies how to click on a text during a replay of a `TextClick`.

Syntax

```
Public Enumeration ClickType
```

Members

Name	Description
Left	A single click with the left mouse button.
Right	A single click with the right mouse button.
Middle	A single click with the middle mouse button.
LeftDouble	A double click with the left mouse button.

Name	Description
Press	Clicking and holding the left mouse button.
Release	Releasing the left mouse button.

Color Class

Description

Describes a color in terms of red, green, blue, and alpha channels.

Inheritance Hierarchy

`Color` does not derive from any class, and no classes derive from it.

Syntax

```
Public Class Color
```

Fields

Name	Description
BLACK	Gets the color that has the ARGB value of #FF000000.
WHITE	Gets the color that has the ARGB value of #FFFFFFFF.
GREEN	Gets the color that has the ARGB value of #FF00FF00.
RED	Gets the color that has the ARGB value of #FFFF0000.
BLUE	Gets the color that has the ARGB value of #FF0000FF.
YELLOW	Gets the color that has the ARGB value of #FFFFFF00.
MAGENTA	Gets the color that has the ARGB value of #FFFF00FF.
CYAN	Gets the color that has the ARGB value of #FF00FFFF.

Properties

Name	Description
R	Gets the value of the red ARGB channel of the color. `Integer`
G	Gets the value of the green ARGB channel of the color. `Integer`
B	Gets the value of the blue ARGB channel of the color. `Integer`
A	Gets the value of the alpha ARGB channel of the color. `Integer`

ItemIdentifier Data Type

Description

Specifies an item in controls such as list boxes, tab controls, or combo boxes. Items can be identified by the following:

- Text (`String`).
- Index (`Integer`).
- A combination of text and index (`String`).

A string value specifies the name of the item. The string can contain wildcard characters. You can also specify the index of the item as a string, in the form `"[n]"`, where n is the numerical index of the item.

Example

The following examples show how you can select an item in a `ListBox`:

```
listbox.select(0)  ' Selects the first item, which has the
index 0.
listbox.select("[0]")  ' Also selects the first item.
listbox.select("apple")   ' Selects the first item with text
"apple".
listbox.select("apple[1]")   'Selects the second item with the
text "apple".
```

ItemPath Data Type

Description

The hierarchical path from the root of the tree to a specific item. You have to always specify the entire hierarchy. Each part in the hierarchy of the item path can either be a name, an index, or a combination of both. You can uniquely identify each item in the hierarchical path by specifying one or more of its attributes.

Items in a tree have the following attributes:

- The hierarchical relationship between the item and other items.
- The title or the caption of the item.
- The 0-based ordinal number of the item in relation to its siblings.

Example

The following examples show how you can identify unique items within a tree by path, index, or a combination of both path and index:

Specification	Item
"/Desktop/ Applications/ Games"	A unique item "Games" under a unique item "Applications" under a unique item "Desktop" at the outer level of the tree.
"/Desktop/[2]/ [4]"	The fifth item under the third item under a unique item "Desktop" at the outer level of the tree.
"/Desktop/ Applications[2]/ Games"	A unique item "Games" under the third item with the text "Applications" under a unique item "Desktop" at the outer level of the tree.

ModifierKeys Enumeration

Description

Specifies the key modifiers that should be pressed during replay.

Syntax

```
Public Enumeration ModifierKeys
```

Members

Name	Description
None	No key modifiers.
Shift	The **Shift** modifier.
Alt	The **Alt** modifier.
Cmd	The **Cmd** modifier. Supported only for desktop browsers on macOS.
Control	The **Ctrl** modifier.
ControlAlt	The **Ctrl+Alt** modifiers.
ShiftControl	The **Shift+Ctrl** modifiers.
ShiftAlt	The **Shift+Alt** modifiers.
ShiftControlAlt	The **Ctrl+Shift+Alt** modifiers.

Example

To follow a link from a Microsoft Word document that you are currently editing, you have to press **Ctrl** and click on the link. You can do this as follows:

```
link.Click(MouseButton.Left, ClickPosition.Center,
ModifierKeys.Control)
```

Note: All three parameters are required.

MouseButton Enumeration

Description

Defines values that specify the buttons on a mouse device.

Syntax

```
Public Enumeration MouseButton
```

Members

Name	Description
Left	The left mouse button. Left = 1.
Middle	The middle mouse button. Middle = 3.

Name	Description
Right	The right mouse button. Right = 2.

Point Class

Description

Specifies a point on the screen.

Inheritance Hierarchy

`Point` does not derive from any class, and no classes derive from it.

Syntax

```
Public Class Point
```

Properties

Name	Description
X	Gets the x coordinate.
Y	Gets the y coordinate.

Range Class

Description

Specifies a range with a minimum and a maximum.

Inheritance Hierarchy

`Range` does not derive from any class, and no classes derive from it.

Syntax

```
Public Class Range
```

Properties

Name	Description
Minimum	Gets the minimum.
Maximum	Gets the maximum.

Rectangle Class

Description

Specifies a rectangle on the screen.

Inheritance Hierarchy

`Rectangle` does not derive from any class, and no classes derive from it.

Syntax

```
Public Class Rectangle
```

Properties

Name	Description
X	Gets the x coordinate of the rectangle.
Y	Gets the y coordinate of the rectangle.
Width	Gets the width of the rectangle.
Height	Gets the height of the rectangle.
Center	Gets the center of the rectangle.
IsEmpty	Determines whether the rectangle is empty.

Methods

Name	Description
Contains(Rectangle)	Determines whether the current `Rectangle` contains the specified `Rectangle`. `Boolean`
Contains(Point)	Determines whether the current `Rectangle` contains the specified `Point`. `Boolean`
Contains(X,Y)	Determines whether the current `Rectangle` contains the specified coordinates. `Boolean`

TextPosition Class

Description

Specifies a position within a text field.

Inheritance Hierarchy

`TextPosition` does not derive from any class, and no classes derive from it.

Syntax

```
Public Class TextPosition
```

Properties

Name	Description
Line	Gets the line.
Column	Gets the column.

TextRange Class

Description

Specifies a range within a text field.

Inheritance Hierarchy

`TextRange` does not derive from any class, and no classes derive from it.

Syntax

```
Public Class TextRange
```

Properties

Name	Description
`StartLine`	Gets the start line of the range.
`StartColumn`	Gets the start column of the range.
`EndLine`	Gets the end line of the range.
`EndColumn`	Gets the end column of the range.

TreeContent Class

Description

Represents the content of a tree control. Every item in a tree is represented as a `TreeNode`.

Inheritance Hierarchy

`TreeContent` does not derive from any class, and no classes derive from it.

Syntax

```
Public Class TreeContent
```

Properties

Name	Description
`RootNodes`	Gets the root nodes of the corresponding tree.

TreeNode Class

Description

Represents a node in the `TreeContent`.

Inheritance Hierarchy

`TreeNode` does not derive from any class, and no classes derive from it.

Syntax

```
Public Class TreeNode
```

Properties

Name	Description
`Children`	Gets the child nodes of the `TreeNode`.

Name	Description
Text	Gets the text of the `TreeNode`.

Flex Class Reference

Lists the available classes for testing Flex controls. You can also customize Silk Test to test custom controls.

For additional information about the classes, see the *Apache Flex Developer's Guide*.

Note: Any known problems with Flex and Flex classes can be reproduced in Silk Test. For instance, dragging and dropping an element that uses the FlexChart class may not work if a chart element is clicked near its edges before you start dragging the element. This results in a failed replay because the drag start coordinates can be incorrect. This is a known issue with dragging and dropping in Flex. For details about known issues with Flex, see the *Apache Flex Release Notes*.

Note: The `FlexTitleWindow` class is recorded and recognized by Silk Test as a `FlexPanel` (although it is defined as a separate class). This is a known issue in Flex. For details about known issues with Flex, see the *Apache Flex Release Notes*.

FlexAccordion Class

Description

An Accordion navigator container has a collection of child containers, but only one of them at a time is visible. It creates and manages navigator buttons (accordion headers), which you use to navigate between the children.

There is one navigator button associated with each child container, and each navigator button belongs to the Accordion container, not to the child. When the user clicks a navigator button, the associated child container is displayed. The transition to the new child uses an animation to make it clear to the user that one child is disappearing and a different one is appearing.

The Accordion container does not extend the ViewStack container, but it implements all the properties, methods, styles, and events of the ViewStack container, such as selectedIndex and selectedChild.

Inheritance Hierarchy

- *FlexContainer*
  - `FlexAccordion`

Syntax

```
'Declaration
Public Class FlexAccordion _
Inherits FlexContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)

Name	Description
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)

Name	Description
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
headerHeight	the height of each accordion header, in pixels.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalGap	the number of pixels between children in the horizontal direction. The default value is 8.
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
paddingBottom	the number of pixels between the container's bottom border and its content area.
paddingTop	the number of pixels between the container's top border and its content area.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChild	a reference to the currently visible child container.
selectedIndex	the zero-based index of the currently visible child container.

Name	Description
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textSelectedColor	the color of selected text. The default value is 0x2B333C.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalGap	the number of pixels between children in the vertical direction.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when the selected child container changes.
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexAdvancedDataGrid Class

Description

Expands on the functionality of the standard DataGrid control to add data visualization features to your Apache Flex application. These features provide greater control of data display, data aggregation, and data formatting.

The AdvancedDataGrid control is like a List control except that it can show more than one column of data, making it suited for showing objects with multiple properties.

Inheritance Hierarchy

- *FlexListBase*
  - FlexAdvancedDataGrid
    - *FlexOLAPDataGrid*

Syntax

```
'Declaration
Public Class FlexAdvancedDataGrid _
Inherits FlexListBase
```

Properties

Name	Description
allowDragSelection	whether drag-selection is enabled. Drag-Selection is the ability to select an item by dragging into it as opposed to normal selection where you can't have the mouse button down when you mouse over the item you want to select.
allowMultipleSelection	whether you can allow more than one item to be selected at the same time. (Inherited from *FlexListBase*)
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alternatingItemColors	the colors to use for the backgrounds of the items in the list. (Inherited from *FlexListBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundDisabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3. (Inherited from *FlexListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnCount	the number of columns to be displayed in a TileList control or items in a HorizontalList control. (Inherited from *FlexListBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexListBase*)
columnWidth	the width of the control's columns. (Inherited from *FlexListBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipField	the name of the field in the data provider items to display as the data tip. (Inherited from *FlexListBase*)
depthColors	the array of colors used for the rows of each level of the navigation tree of the AdvancedDataGrid control, in descending order. The default value is undefined.

Name	Description
disabledColor	the color of text in the component if it is disabled.
displayItemsExpanded	whether the navigation tree is expanded to show all items.
editable	whether the user can edit items in the data provider.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexListBase*)
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4. (Inherited from *FlexListBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
headerColors	an array of two colors used to draw the header background gradient.
headerHeight	the height of the header cell of the column, in pixels.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalGridLineColor	the color of the horizontal grid lines.
iconField	the name of the field in the data provider object that determines what to display as the icon. (Inherited from *FlexListBase*)
id	the ID of the component. (Inherited from *FlexObject*)
indentation	the indentation for each node of the navigation tree, in pixels. The default value is 17.
labelField	the name of the field in the data provider items to display as the label. (Inherited from *FlexListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexListBase*)
lockedColumnCount	the index of the first column in the control that scrolls. (Inherited from *FlexListBase*)
lockedRowCount	the index of the first row in the control that scrolls. (Inherited from *FlexListBase*)
minColumnWidth	the minimum width of the columns, in pixels.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexListBase*)
numRows	the total number of rows of data available. (Inherited from *FlexListBase*)
openDuration	the length of an open or close transition for the navigation tree, in milliseconds. The default value is 250.
paddingBottom	the number of pixels between the bottom of the row and the bottom of the renderer in the row.
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2.
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0.
paddingTop	the number of pixels between the top of the row and the top of the renderer in the row.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizableColumns	whether the user can change the size of the columns.
rollOverColor	the color of the background of a renderer when the user rolls over it. The default value is 0xEEFEE6. (Inherited from *FlexListBase*)
rowCount	the number of rows to be displayed. (Inherited from *FlexListBase*)
rowHeight	the height of the rows in pixels. (Inherited from *FlexListBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	whether the list shows selected items as selected. (Inherited from *FlexListBase*)
selectedCells	an Array of cell locations as row and column indices.
selectedIndex	the index in the data provider of the selected item. (Inherited from *FlexListBase*)
selectedIndices	an array of indices in the data provider of the selected items. (Inherited from *FlexListBase*)
selectedItem	a reference to the selected item in the data provider. (Inherited from *FlexListBase*)

Name	Description
selectedItems	an array of references to the selected items in the data provider. (Inherited from *FlexListBase*)
selectionColor	the color of the background of a renderer when the user selects it. The default value is 0x7FCEFF. (Inherited from *FlexListBase*)
selectionDisabledColor	the color of the background of a renderer when the component is disabled. The default value is 0xDDDDDD. (Inherited from *FlexListBase*)
selectionMode	the selection mode of the control.
showDataTips	whether dataTips are displayed for text in the rows. (Inherited from *FlexListBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C. (Inherited from *FlexListBase*)
textSelectedColor	the color of the text of a renderer when the user selects it. The default value is 0x2B333C. (Inherited from *FlexListBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useRollOver	whether items are highlighted as the mouse rolls over them. (Inherited from *FlexListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
variableRowHeight	whether the individual rows can have different height. (Inherited from *FlexListBase*)
verticalAlign	the vertical alignment of a renderer in a row. (Inherited from *FlexListBase*)
verticalGridLineColor	the color of the vertical grid lines. The default value is 0x666666.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)

Name	Description
wordWrap	whether text in the row should be word wrapped. (Inherited from *FlexListBase*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes or collapses a AdvancedDataGrid branch.
ColumnGroupedADGHeaderShift	Dispatched when the user releases the mouse button on a column header after having dragged the column to a new location resulting in shifting the column to a new index.
ColumnStretch	Dispatched when a user changes the width of a column, indicating that the amount of data displayed in that column may have changed.
Deselect	Defines the value of the type property of the event object for an event that is dispatched when a previously selected item is deselected. (Inherited from *FlexListBase*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Defines the value of the type property of the ListEvent object for an itemDoubleClick event, which indicates that the user double-clicked the mouse over a visual item in the control. (Inherited from *FlexListBase*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexListBase*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexListBase*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexListBase*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCellData	Returns data for a cell in the grid.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetGroupedItemChildrenCount	Returns the number of children within the first item of a group.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetRowData	Returns data for the row of the item in the data provider.
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexListBase*)
HeaderClick	Dispatched when the user releases the mouse button on a column header to request the control to sort the grid contents based on the contents of the column.
HeaderShift	Dispatched when the user releases the mouse button on a column header after having dragged the column to a new location resulting in shifting the column to a new index.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)

Name	Description
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsGroupedItem	Returns a value that specifies whether the item is a member of a group.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexScrollBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
MultiSelect	Defines the value of the type property of the event object for an event that is dispatched when an item is selected as part of an action that selects multiple items. (Inherited from *FlexListBase*)
Open	Opens or expands an AdvancedDataGrid branch.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event. (Inherited from *FlexListBase*)
ScrollToIndex	Ensures that the data provider item at the given index is visible. (Inherited from *FlexListBase*)
Select	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)

Name	Description
SelectIndex	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexAlert Class

Description

A pop-up dialog box that can contain a message, a title, buttons (any combination of OK, Cancel, Yes, and No) and an icon.

The Alert control is modal, which means it will retain focus until the user closes it.

Inheritance Hierarchy

- *FlexPanel*
  - FlexAlert

Syntax

```
'Declaration
Public Class FlexAlert _
Inherits FlexPanel
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)

Name	Description
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
defaultButtonFlag	A bitflag that contains either Alert.OK, Alert.CANCEL, Alert.YES, or Alert.NO to specify the default button. The default value is Alert.OK.
dropShadowEnabled	whether the Panel container's drop shadow is visible. The default value is true. (Inherited from *FlexPanel*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
footerColors	the array of two colors used to draw the footer (area for the ControlBar container) background. (Inherited from *FlexPanel*)
headerColors	the array of two colors used to draw the header. (Inherited from *FlexPanel*)
headerHeight	the height of the header. The default value is based on the style of the title text. (Inherited from *FlexPanel*)

Name	Description
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
status	the text in the status area of the title bar. The default value is "". (Inherited from *FlexPanel*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	The text to display in this alert dialog box. The default value is "".
themeColor	the theme color of a component. (Inherited from *FlexObject*)

Name	Description
title	the title or caption displayed in the title bar. The default value is "". (Inherited from *FlexPanel*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a

Name	Description
	timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexApplication Class

Description
A default, or Application, container that lets you start adding content to your application without explicitly defining another container.

Inheritance Hierarchy
- *FlexContainer*
  - FlexApplication
    - *FlexWindowedApplication*

Syntax
```
'Declaration
Public Class FlexApplication _
Inherits FlexContainer
```

Properties
Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)

Name	Description
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)

Name	Description
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
url	the URL from which this Application's SWF file was loaded.
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexAreaChart Class

Description

Represents data as an area bounded by a line connecting the values in the data. The AreaChart control can be used to represent different variations, including simple areas, stacked, 100% stacked, and high/low.

Inheritance Hierarchy

- *FlexCartesianChart*
  - FlexAreaChart

Syntax

```
'Declaration
Public Class FlexAreaChart _
Inherits FlexCartesianChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)
description	a short description of the data in the chart. (Inherited from *FlexChart*)
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)

Name	Description
gutterBottom	The size of the region, in pixels, between the bottom of the chart data area and the bottom of the chart control. (Inherited from *FlexCartesianChart*)
gutterLeft	The size of the region, in pixels, between the left of the chart data area and the left of the chart control. (Inherited from *FlexCartesianChart*)
gutterRight	The size of the region, in pixels, between the right side of the chart data area and the outside of the chart control. (Inherited from *FlexCartesianChart*)
gutterTop	The size of the region, in pixels, between the top of the chart data area and the top of the chart control. (Inherited from *FlexCartesianChart*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAxisRatio	the height limit of the horiztonal axis. (Inherited from *FlexCartesianChart*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)

Name	Description
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexCartesianChart*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
verticalAxisRatio	the width limit of the vertical axis. (Inherited from *FlexCartesianChart*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexAreaSeries Class

Description
Defines a data series for an AreaChart control.

Inheritance Hierarchy

- *FlexChartSeries*
  - FlexAreaSeries

Syntax

```
'Declaration
Public Class FlexAreaSeries _
Inherits FlexChartSeries
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
form	the boundary type for the area.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
minField	the field of the dataProvider that determines the bottom boundary of the area.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)

Name	Description
radius	the radius, in pixels, of the chart elements for the data points.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
xField	the field of the data provider that determines the position of the data points on the horizontal axis.
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
yField	the field of the data provider that determines the position of the data point on the vertical axis.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)

Name	Description
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexAxisRenderer Class

Description

Describe the horizontal and vertical axes of a chart. An axis is responsible for rendering the labels, tick marks, and title along its length.

AxisRenderer objects inherit some of their visual properties from the enclosing chart object. The text format of the labels and title defaults to the CSS text properties of the renderer. You can control the formatting of the axis title separately by specifying a axisTitleStyleName, either on the AxisRenderer or on the enclosing chart.

Inheritance Hierarchy

- *FlexObject*
  - FlexAxisRenderer

Syntax

```
'Declaration
Public Class FlexAxisRenderer _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
canDropLabels	whether the AxisRenderer should drop labels as necessary to lay out correctly.
canStagger	whether to stagger labels on two label rows. Use this setting to minimize the space required for the labels. The default value is true, which staggers the labels.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)

Name	Description
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
heightLimit	the maximum amount of space, in pixels, that an axis renderer will take from a chart.
id	the ID of the component. (Inherited from *FlexObject*)
labelGap	the gap between the end of the tick marks and the top of the labels, in pixels. The default value is 3.
labelRotation	the label rotation.
lastVisibleRow	the index of the last visible child.
length	the length of the axis, in screen coordinates.
minorTickLength	the length of the minor tick marks on the axis, in pixels. The default value is 0.
minorTickPlacement	a value where to draw the minor tick marks. Values include: *inside*, *outside*, *cross*, and *none*.
minorTicks	an array that specifies where Flex draws the minor tick marks along the axis. Each array element contains a value between 0 and 1.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
placement	the side of the chart that the axisRenderer appears on.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)

Name	Description
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
showLabels	whether labels appear along the axis. The default value is true.
showLine	whether to display the axis. The default value is true.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
tickLength	the length of the tick marks on the axis, in pixels. The default value is 3.
tickPlacement	a value where to draw the tick marks. Values include: *inside*, *outside*, *cross*, and *none*.
ticks	an array that specifies where Flex draws the tick marks along the axis. Each array element contains a value between 0 and 1.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexBarChart Class

Description
Represents data as a series of horizontal bars whose length is determined by values in the data. A BarChart control can represent different chart variations, including simple bars, clustered bars, stacked, 100% stacked, and high/low.

Inheritance Hierarchy

- *FlexCartesianChart*
  - FlexBarChart

Syntax

```
'Declaration
Public Class FlexBarChart _
Inherits FlexCartesianChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
barWidthRatio	how wide to draw the bars relative to the category width, as a percentage in the range of 0 to 1.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)
description	a short description of the data in the chart. (Inherited from *FlexChart*)
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)

Name	Description
gutterBottom	The size of the region, in pixels, between the bottom of the chart data area and the bottom of the chart control. (Inherited from *FlexCartesianChart*)
gutterLeft	The size of the region, in pixels, between the left of the chart data area and the left of the chart control. (Inherited from *FlexCartesianChart*)
gutterRight	The size of the region, in pixels, between the right side of the chart data area and the outside of the chart control. (Inherited from *FlexCartesianChart*)
gutterTop	The size of the region, in pixels, between the top of the chart data area and the top of the chart control. (Inherited from *FlexCartesianChart*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAxisRatio	the height limit of the horiztonal axis. (Inherited from *FlexCartesianChart*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maxBarWidth	how wide to draw the bars, in pixels.
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)

Name	Description
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexCartesianChart*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
verticalAxisRatio	the width limit of the vertical axis. (Inherited from *FlexCartesianChart*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexBarSeries Class

Description
Defines a data series for a BarChart control.

Inheritance Hierarchy

- *FlexChartSeries*
  - FlexBarSeries

Syntax

```
'Declaration
Public Class FlexBarSeries _
Inherits FlexChartSeries
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
barWidthRatio	the width to render the bars relative to the category width.

Name	Description
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
maxBarWidth	the width of the bars, in pixels.
minField	the field of the data provider that determines the bottom of each bar.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)
offset	the amount to offset the center of the bars from the center of the available space, relative to the category width.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)

Name	Description
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
xField	the field of the data provider that determines the x-axis location of the top of each bar.
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Name	Description
yField	the field of the data provider that determines the y-axis location of the bottom of each bar in the chart.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)

Name	Description
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexBox Class

Description
A Box container lays out its children in a single vertical column or a single horizontal row.

Inheritance Hierarchy

- *FlexContainer*
  - FlexBox
    - *FlexDividedBox*
    - *FlexNavigationBar*

Syntax

```
'Declaration
Public Class FlexBox _
Inherits FlexContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
direction	whether to use vertical (default) or horizontal layout.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAlign	the horizontal alignment of children in the container.
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAlign	the vertical alignment of children in the container.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Name	Description
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexBubbleChart Class

Description

Represents data with three values for each data point. Each data point is defined by a value determining its position along the horizontal axis, a value determining its position along the vertical axis, and a value determining the size of the chart element, relative to the other data points on the chart.

Inheritance Hierarchy

- *FlexCartesianChart*
  - FlexBubbleChart

Syntax

```
'Declaration
Public Class FlexBubbleChart _
Inherits FlexCartesianChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)

Name	Description
description	a short description of the data in the chart. (Inherited from *FlexChart*)
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)
gutterBottom	The size of the region, in pixels, between the bottom of the chart data area and the bottom of the chart control. (Inherited from *FlexCartesianChart*)
gutterLeft	The size of the region, in pixels, between the left of the chart data area and the left of the chart control. (Inherited from *FlexCartesianChart*)
gutterRight	The size of the region, in pixels, between the right side of the chart data area and the outside of the chart control. (Inherited from *FlexCartesianChart*)
gutterTop	The size of the region, in pixels, between the top of the chart data area and the top of the chart control. (Inherited from *FlexCartesianChart*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAxisRatio	the height limit of the horiztonal axis. (Inherited from *FlexCartesianChart*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to

Name	Description
	false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)

Name	Description
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexCartesianChart*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAxisRatio	the width limit of the vertical axis. (Inherited from *FlexCartesianChart*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexBubbleSeries Class

Description
Defines a data series for a BubbleChart control.

Inheritance Hierarchy

- *FlexChartSeries*
  - FlexBubbleSeries

Syntax

```
'Declaration
Public Class FlexBubbleSeries _
Inherits FlexChartSeries
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
maxRadius	The radius of the largest item renderered in this series.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list,

Name	Description
	use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
radiusField	the field of the data provider that determines the radius of each symbol, relative to the other data points in the chart.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
xField	the field of the data provider that determines the x-axis location of each data point.
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
yField	the field of the data provider that determines the y-axis location of each data point.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexButton Class

Description

The Button control is a commonly used rectangular button. Button controls look like they can be pressed. They can have a text label, an icon, or both on their face.

Buttons typically use event listeners to perform an action when the user selects the control. When a user clicks the mouse on a Button control, and the Button control is enabled, it dispatches a click event and a buttonDown event. A button always dispatches events such as the mouseMove, mouseOver, mouseOut, rollOver, rollOut, mouseDown, and mouseUp events whether enabled or disabled.

You can customize the look of a Button control and change its functionality from a push button to a toggle button. You can change the button appearance by using a skin for each of the button's states.

The label of a Button control uses a bold typeface. If you embed a font that you want to use for the label of the Button control, you must embed the bold typeface.

Inheritance Hierarchy

- *FlexObject*
  - FlexButton
    - *FlexCheckBox*
    - *FlexPopUpButton*
    - *FlexRadioButton*

Syntax

```
'Declaration
Public Class FlexButton _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. The default value is 0x0B333C.
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alphas used for the background fill of controls.
fillColors	the colors used to tint the background of the control.
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels.
fontStyle	whether the text is italic font.
fontWeight	whether the text is boldface.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the Button control.
labelPlacement	the orientation of the label in relation to a specified icon.

Name	Description
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	whether a toggle button is toggled on (true) or off (false).
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container.
textDecoration	whether the text is underlined.
textIndent	the offset of the first line of text from the left side of the container, in pixels. The default value is 0.
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C.
textSelectedColor	the text color of the label as the user presses it. The default value is 0x000000.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toggle	whether a Button is in a toggle state or not.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexButtonBar Class

Description
Defines a horizontal or vertical group of logically related push buttons with a common look and navigation.

Inheritance Hierarchy

- *FlexNavigationBar*
  - FlexButtonBar
    - *FlexToggleButtonBar*

Syntax

```
'Declaration
Public Class FlexButtonBar _
Inherits FlexNavigationBar
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)

Name	Description
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
direction	whether to use vertical (default) or horizontal layout. (Inherited from *FlexBox*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAlign	the horizontal alignment of children in the container. (Inherited from *FlexBox*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to

Name	Description
	false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the active navigation item, where the first item is at an index of 0. The default value is -1. (Inherited from *FlexNavigationBar*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAlign	the vertical alignment of children in the container. (Inherited from *FlexBox*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Defines the value of the type property of the event object for an itemClick event. (Inherited from *FlexNavigationBar*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexNavigationBar*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexCandlestickChart Class

Description

Represents financial data as a series of candlesticks representing the high, low, opening, and closing values of a data series.

The top and bottom of the vertical line in each candlestick represent the high and low values for the datapoint, while the top and bottom of the filled box represents the opening and closing values. Each candlestick is filled differently depending on whether the closing value for the datapoint is higher or lower than the opening value.

Inheritance Hierarchy

- *FlexCartesianChart*
  - FlexCandlestickChart

Syntax

```
'Declaration
Public Class FlexCandlestickChart _
Inherits FlexCartesianChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)

Name	Description
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)
columnWidthRatio	a ratio of how wide to draw the candlesticks relative to the horizontal axis's category widths, as a percentage in the range of 0 to 1.
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)
description	a short description of the data in the chart. (Inherited from *FlexChart*)
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)
gutterBottom	The size of the region, in pixels, between the bottom of the chart data area and the bottom of the chart control. (Inherited from *FlexCartesianChart*)

Name	Description
gutterLeft	The size of the region, in pixels, between the left of the chart data area and the left of the chart control. (Inherited from *FlexCartesianChart*)
gutterRight	The size of the region, in pixels, between the right side of the chart data area and the outside of the chart control. (Inherited from *FlexCartesianChart*)
gutterTop	The size of the region, in pixels, between the top of the chart data area and the top of the chart control. (Inherited from *FlexCartesianChart*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAxisRatio	the height limit of the horiztonal axis. (Inherited from *FlexCartesianChart*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maxColumnWidth	how wide to draw the candlesticks, in pixels.
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)

Name	Description
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexCartesianChart*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAxisRatio	the width limit of the vertical axis. (Inherited from *FlexCartesianChart*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object,

Name	Description
	for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexCandlestickSeries Class

Description

Represents financial data as a series of candlesticks representing the high, low, opening, and closing values of a data series.

The top and bottom of the vertical line in each candlestick represent the high and low values for the datapoint, while the top and bottom of the filled box represent the opening and closing values. Each candlestick is filled differently depending on whether the closing value for the datapoint is higher or lower than the opening value.

Inheritance Hierarchy

- *FlexHLOCSeriesBase*
  - FlexCandlestickSeries

Syntax

```
'Declaration
Public Class FlexCandlestickSeries _
Inherits FlexHLOCSeriesBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
columnWidthRatio	the width of elements relative to the category width. (Inherited from *FlexHLOCSeriesBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
maxColumnWidth	the width of the elements, in pixels. (Inherited from *FlexHLOCSeriesBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)

Name	Description
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
xField	the field of the data provider that determines the x-axis location of the element. (Inherited from *FlexHLOCSeriesBase*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)

Name	Description
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexCanvas Class

Description

A Canvas layout container defines a rectangular region in which you place child containers and controls. It is the only container that lets you explicitly specify the location of its children within the container by using the x and y properties of each child.

Flex sets the children of a Canvas layout container to their preferred width and preferred height. You may override the value for a child's preferred width by setting its width property to either a fixed pixel value or a percentage of the container size. You can set the preferred height in a similar manner.

If you use percentage sizing inside a Canvas container, some of your components may overlap. If this is not the effect you want, plan your component positions and sizes carefully.

Inheritance Hierarchy

- *FlexContainer*
  - FlexCanvas

Syntax

```
'Declaration
Public Class FlexCanvas _
Inherits FlexContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)

Name	Description
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)

Name	Description
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexCartesianChart Class

Description
A base class for the common chart types. CartesianChart defines the basic layout behavior of the standard rectangular, two-dimensional charts.

Inheritance Hierarchy

- *FlexChart*
  - FlexCartesianChart
    - *FlexAreaChart*
    - *FlexBarChart*
    - *FlexBubbleChart*
    - *FlexCandlestickChart*
    - *FlexColumnChart*
    - *FlexHLOCChart*
    - *FlexLineChart*
    - *FlexPlotChart*

Syntax

```
'Declaration
Public Class FlexCartesianChart _
Inherits FlexChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)

Name	Description
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)
description	a short description of the data in the chart. (Inherited from *FlexChart*)
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)
gutterBottom	The size of the region, in pixels, between the bottom of the chart data area and the bottom of the chart control.
gutterLeft	The size of the region, in pixels, between the left of the chart data area and the left of the chart control.
gutterRight	The size of the region, in pixels, between the right side of the chart data area and the outside of the chart control.
gutterTop	The size of the region, in pixels, between the top of the chart data area and the top of the chart control.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAxisRatio	the height limit of the horiztonal axis.
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to

Name	Description
	false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)

Name	Description
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAxisRatio	the width limit of the vertical axis.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexChart Class

Description

The link between the Flex component architecture and the DualStyleObject architecture. It extends the Flex UIComponent base class, but contains DualStyleObject classes.

You typically do not use the ChartBase class directly. Instead you use one of its subclasses, such as PlotChart or BubbleChart. It acts as the base class for the common chart types provided in the mx.charts package.

This class defines a number of CSS styles and properties that provide easy access to the more common features of the framework.

A chart's minimum size is 20 x 20 pixels. A chart's maximum size is unbounded. A chart's preferred size is 400 x 400 pixels.

Inheritance Hierarchy

- *FlexObject*
  - FlexChart
    - *FlexCartesianChart*
    - *FlexPieChart*

Syntax

```
'Declaration
Public Class FlexChart _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered.
columnNames	a list containing the names of all columns in the data.
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart.
description	a short description of the data in the chart.
dragEnabled	whether you can drag items out of this chart and drop them on other controls.
dropEnabled	whether dragged items can be dropped onto the chart.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child.
maximumDataTipCount	the maximum number of datatips a chart will show.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart.
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
paddingBottom	the number of pixels between the chart's bottom border and its content area.
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2.
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0.
paddingTop	the number of pixels between the chart's top border and its content area.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)

Name	Description
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item.
selectedChartItems	an array of all the selected ChartItem objects in the chart.
selectionMode	whether ChartItem objects can be selected.
showDataTips	whether DataTip controls for the chart show.
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event.
DragDrop	Defines the value of the type property of the event object for a dragDrop event.
DragStart	Defines the value of the type property of the event object for a dragStart event.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.

Name	Description
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexChartLegend Class

Description
Adds a legend to your charts, where the legend displays the label for each data series in the chart and a key showing the chart element for the series.

Inheritance Hierarchy

- *FlexObject*
  - FlexChartLegend

Syntax

```
'Declaration
Public Class FlexChartLegend _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontWeight	whether the text is boldface.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text that displays next to the legend element.
labelPlacement	the label placement of the legend element. Valid values include: *top*, *bottom*, *right*, and *left*.
lastVisibleRow	the index of the last visible child.
markerAspectRatio	the aspect ratio for the marker associated with this legend item.
markerHeight	the height of the legend element.
markerWidth	the width of the legend element.

Name	Description
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2.
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
verticalGap	the number of pixels between children in the vertical direction. The default value depends on the component class; if not overriden for the class, the default value is 6.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
LegendItemClick	Dispatched when the user clicks on a LegendItem in the Legend control, which indicates that the user clicked the mouse button over a legend item.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexChartSeries Class

Description

The base class for the classes that let you specify a data series for a chart control.

You use the subclasses of the Series class with the associated chart control. You can use a Series class to specify the fill pattern and stroke characteristics for the chart elements that are associated with the data series.

Inheritance Hierarchy

- *FlexObject*
  - FlexChartSeries
    - *FlexAreaSeries*
    - *FlexBarSeries*
    - *FlexBubbleSeries*
    - *FlexColumnSeries*
    - *FlexHLOCSeriesBase*
    - *FlexLineSeries*
    - *FlexPieSeries*
    - *FlexPlotSeries*

Syntax

```
'Declaration
Public Class FlexChartSeries _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)

Name	Description
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series.

Name	Description
selectedIndices	an array of indexes of the selected items in the data provider of the series.
selectedItem	the chart item that is selected in the series.
selectedItems	an array of chart items that are selected in the series.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart.
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart.
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexCheckBox Class

Description

The CheckBox control consists of an optional label and a small box that can contain a check mark or not.

You can place the optional text label to the left, right, top, or bottom of the CheckBox. When a user clicks a CheckBox control or its associated text, the CheckBox control changes its state from checked to unchecked or from unchecked to checked. CheckBox controls gather a set of true or false values that are not mutually exclusive.

Inheritance Hierarchy

- *FlexButton*
  - `FlexCheckBox`

Syntax

```
'Declaration
Public Class FlexCheckBox _
Inherits FlexButton
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. The default value is 0x0B333C. (Inherited from *FlexButton*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3. (Inherited from *FlexButton*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alphas used for the background fill of controls. (Inherited from *FlexButton*)
fillColors	the colors used to tint the background of the control. (Inherited from *FlexButton*)
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4. (Inherited from *FlexButton*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexButton*)
fontSize	the height of the text, in pixels. (Inherited from *FlexButton*)
fontStyle	whether the text is italic font. (Inherited from *FlexButton*)
fontWeight	whether the text is boldface. (Inherited from *FlexButton*)

Name	Description
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the Button control. (Inherited from *FlexButton*)
labelPlacement	the orientation of the label in relation to a specified icon. (Inherited from *FlexButton*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	whether a toggle button is toggled on (true) or off (false). (Inherited from *FlexButton*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexButton*)
textDecoration	whether the text is underlined. (Inherited from *FlexButton*)
textIndent	the offset of the first line of text from the left side of the container, in pixels. The default value is 0. (Inherited from *FlexButton*)

Name	Description
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C. (Inherited from *FlexButton*)
textSelectedColor	the text color of the label as the user presses it. The default value is 0x000000. (Inherited from *FlexButton*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toggle	whether a Button is in a toggle state or not. (Inherited from *FlexButton*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexButton*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout

Name	Description
	parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexColorPicker Class

Description

Provides a way for a user to choose a color from a swatch list.

The default mode of the component shows a single swatch in a square button. When the user clicks the swatch button, the swatch panel appears and displays the entire swatch list.

Inheritance Hierarchy

- *FlexComboBase*
  - FlexColorPicker

Syntax

```
'Declaration
Public Class FlexColorPicker _
Inherits FlexComboBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)

Name	Description
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. (Inherited from *FlexComboBase*)
backgroundColor	the background color of a component. (Inherited from *FlexComboBase*)
backgroundImage	the background image of a component. (Inherited from *FlexComboBase*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexComboBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. (Inherited from *FlexComboBase*)
columnCount	the number of columns in the swatch grid. The default value is 20.
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the background color of the component when it is disabled. (Inherited from *FlexComboBase*)
dropShadowEnabled	whether the ComboBase container's drop shadow is visible. The default value is true. (Inherited from *FlexComboBase*)
editable	whether the control is editable, which lets the user directly type entries that are not specified in the dataProvider, or not editable, which requires the user select from the items in the dataProvider. (Inherited from *FlexComboBase*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alphas used for the background fill of controls. (Inherited from *FlexComboBase*)
fillColors	the colors used to tint the background of the control. (Inherited from *FlexComboBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexComboBase*)
fontSize	the height of the text, in pixels. (Inherited from *FlexComboBase*)
fontStyle	whether the text is italic font. (Inherited from *FlexComboBase*)

Name	Description
fontWeight	whether the text is boldface. (Inherited from *FlexComboBase*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
highlightColor	the highlight color of the drop-down list. (Inherited from *FlexComboBase*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
restrict	the set of characters that a user can or cannot enter into the text field. (Inherited from *FlexComboBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedColor	the value of the currently selected color in the SwatchPanel object.
selectedIndex	the index in the data provider of the selected item. (Inherited from *FlexComboBase*)
showTextField	whether to show the text box that displays the color label or hexadecimal color value. The default value is true.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	the contents of the text field. (Inherited from *FlexComboBase*)
textAlign	the alignment of text within a container. Possible values are:left, right, or center. (Inherited from *FlexComboBase*)

Name	Description
textDecoration	whether the text is underlined. (Inherited from *FlexComboBase*)
textIndent	the offset of first line of text from the left side of the container, in pixels. The default value is 0. (Inherited from *FlexComboBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when the selected color changes as a result of user interaction.
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Defines the value of the type property of the event object for a close event. (Inherited from *FlexComboBase*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Creates an Event object that contains information about text events. Event objects are passed as parameters to event listeners. (Inherited from *FlexComboBase*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
Open	Defines the value of the type property of the event object for a open event. (Inherited from *FlexComboBase*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event. (Inherited from *FlexComboBase*)
SelectText	Lets you track selection within a text field. (Inherited from *FlexComboBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexComboBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexColumnChart Class

Description

Represents data as a series of vertical columns whose height is determined by values in the data.

You can use the ColumnChart to represent a variety of different charts including simple columns, clustered columns, stacked, 100% stacked, and high/low.

Inheritance Hierarchy

- *FlexCartesianChart*
  - FlexColumnChart

Syntax

```
'Declaration
Public Class FlexColumnChart _
Inherits FlexCartesianChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)
columnWidthRatio	a ratio of wide to draw the columns relative to the category width, as a percentage in the range of 0 to 1.
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)
description	a short description of the data in the chart. (Inherited from *FlexChart*)
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)

Name	Description
gutterBottom	The size of the region, in pixels, between the bottom of the chart data area and the bottom of the chart control. (Inherited from *FlexCartesianChart*)
gutterLeft	The size of the region, in pixels, between the left of the chart data area and the left of the chart control. (Inherited from *FlexCartesianChart*)
gutterRight	The size of the region, in pixels, between the right side of the chart data area and the outside of the chart control. (Inherited from *FlexCartesianChart*)
gutterTop	The size of the region, in pixels, between the top of the chart data area and the top of the chart control. (Inherited from *FlexCartesianChart*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAxisRatio	the height limit of the horiztonal axis. (Inherited from *FlexCartesianChart*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maxColumnWidth	how wide to draw the columns, in pixels.
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)

Name	Description
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexCartesianChart*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
verticalAxisRatio	the width limit of the vertical axis. (Inherited from *FlexCartesianChart*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexColumnSeries Class

Description
Defines a data series for a ColumnChart control.

Inheritance Hierarchy

- *FlexChartSeries*
  - FlexColumnSeries

Syntax

```
'Declaration
Public Class FlexColumnSeries _
Inherits FlexChartSeries
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
columnWidthRatio	the width of columns relative to the category width.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
maxColumnWidth	the width of the columns, in pixels.
minField	the field of the data provider that determines the y-axis location of the bottom of a column.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)
offset	how far to offset the center of the columns from the center of the available space, relative to the category width.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)

Name	Description
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
sortOnXField	whether the columns are sorted from left to right before rendering.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
xField	the field of the data provider that determines the x-axis location of the column.

Name	Description
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
yField	the field of the data provider that determines the y-axis location of the top of a column.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)

Name	Description
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexComboBase Class

Description

The base class for controls that display text in a text field and have a button that causes a drop-down list to appear where the user can choose which text to display.

Inheritance Hierarchy

- *FlexObject*
  - FlexComboBase
    - *FlexColorPicker*
    - *FlexComboBox*
    - *FlexDateField*

Syntax

```
'Declaration
Public Class FlexComboBase _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style.
backgroundColor	the background color of a component.
backgroundImage	the background image of a component.
backgroundSize	the percentage to change the image size to for the specified backgroundImage.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label.

Name	Description
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the background color of the component when it is disabled.
dropShadowEnabled	whether the ComboBase container's drop shadow is visible. The default value is true.
editable	whether the control is editable, which lets the user directly type entries that are not specified in the dataProvider, or not editable, which requires the user select from the items in the dataProvider.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alphas used for the background fill of controls.
fillColors	the colors used to tint the background of the control.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels.
fontStyle	whether the text is italic font.
fontWeight	whether the text is boldface.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
highlightColor	the highlight color of the drop-down list.
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is

Name	Description
	NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
restrict	the set of characters that a user can or cannot enter into the text field.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index in the data provider of the selected item.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	the contents of the text field.
textAlign	the alignment of text within a container. Possible values are:left, right, or center.
textDecoration	whether the text is underlined.
textIndent	the offset of first line of text from the left side of the container, in pixels. The default value is 0.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Defines the value of the type property of the event object for a close event.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Creates an Event object that contains information about text events. Event objects are passed as parameters to event listeners.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
Open	Defines the value of the type property of the event object for a open event.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event.
SelectText	Lets you track selection within a text field.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexComboBox Class

Description

Contains a drop-down list from which the user can select a single value.

The ComboBox can be editable, in which case the user can type entries into the TextInput portion of the ComboBox that are not in the list.

Inheritance Hierarchy

- *FlexComboBase*
  - FlexComboBox

Syntax

```
'Declaration
Public Class FlexComboBox _
Inherits FlexComboBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alternatingItemColors	the set of BackgroundColors for drop-down list rows in an alternating pattern.
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. (Inherited from *FlexComboBase*)
backgroundColor	the background color of a component. (Inherited from *FlexComboBase*)
backgroundImage	the background image of a component. (Inherited from *FlexComboBase*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexComboBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. (Inherited from *FlexComboBase*)
columnNames	a list containing the names of all columns in the data.

Name	Description
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the background color of the component when it is disabled. (Inherited from *FlexComboBase*)
dropShadowEnabled	whether the ComboBase container's drop shadow is visible. The default value is true. (Inherited from *FlexComboBase*)
editable	whether the control is editable, which lets the user directly type entries that are not specified in the dataProvider, or not editable, which requires the user select from the items in the dataProvider. (Inherited from *FlexComboBase*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alphas used for the background fill of controls. (Inherited from *FlexComboBase*)
fillColors	the colors used to tint the background of the control. (Inherited from *FlexComboBase*)
firstVisibleRow	the index of the first visible child.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexComboBase*)
fontSize	the height of the text, in pixels. (Inherited from *FlexComboBase*)
fontStyle	whether the text is italic font. (Inherited from *FlexComboBase*)
fontWeight	whether the text is boldface. (Inherited from *FlexComboBase*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
highlightColor	the highlight color of the drop-down list. (Inherited from *FlexComboBase*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
restrict	the set of characters that a user can or cannot enter into the text field. (Inherited from *FlexComboBase*)
rollOverColor	the rollOverColor of the drop-down list.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index in the data provider of the selected item. (Inherited from *FlexComboBase*)
selectionColor	the selectionColor of the drop-down list.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	the contents of the text field. (Inherited from *FlexComboBase*)
textAlign	the alignment of text within a container. Possible values are:left, right, or center. (Inherited from *FlexComboBase*)
textDecoration	whether the text is underlined. (Inherited from *FlexComboBase*)
textIndent	the offset of first line of text from the left side of the container, in pixels. The default value is 0. (Inherited from *FlexComboBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Defines the value of the type property of the event object for a close event. (Inherited from *FlexComboBase*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Creates an Event object that contains information about text events. Event objects are passed as parameters to event listeners. (Inherited from *FlexComboBase*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
Open	Defines the value of the type property of the event object for a open event. (Inherited from *FlexComboBase*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event. (Inherited from *FlexComboBase*)
Select	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected.
SelectIndex	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected.
SelectText	Lets you track selection within a text field. (Inherited from *FlexComboBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexComboBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexContainer Class

Description

An abstract base class for components that controls the layout characteristics of child components.

You do not create an instance of Container in an application. Instead, you create an instance of one of Container's subclasses, such as Canvas or HBox.

The Container class contains the logic for scrolling, clipping, and dynamic instantiation. It contains methods for adding and removing children and the logic for drawing the background and borders of containers.

Inheritance Hierarchy

- *FlexObject*
  - FlexContainer
    - *FlexAccordion*
    - *FlexApplication*
    - *FlexBox*
    - *FlexCanvas*
    - *FlexForm*
    - *FlexFormItem*
    - *FlexPanel*
    - *FlexViewStack*
    - *FlexWindow*

Syntax

```
'Declaration
Public Class FlexContainer _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0.
backgroundColor	the background color of a component.
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined.
backgroundImage	the background image of a component. This can be an absolute or relative URL or class.
backgroundSize	the percentage to change the image size to for the specified backgroundImage.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data.
creationIndex	the order to instantiate and draw the children of the container.
creationPolicy	the child creation policy for this Container.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components.
fillColors	the colors used to tint the background fill of the component.
firstVisibleRow	the index of the first visible child.

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels.
fontStyle	whether the text is italic font.
fontWeight	whether the text is boldface.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container.
lastVisibleRow	the index of the last visible child.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container.
numColumns	the total number of columns of data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event.
DragDrop	Defines the value of the type property of the event object for a dragDrop event.
DragStart	Defines the value of the type property of the event object for a dragStart event.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance.
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexContainerMovieClip Class

Description

Container components created in Adobe Flash CS3 Professional for use in Flex are subclasses of the mx.flash.ContainerMovieClip class.

You can use a subclass of ContainerMovieClip as a Flex container, it can hold children, and it can respond to events, define view states and transitions, and work with effects in the same way as can any Flex component.

A Flash container can only have a single Flex child. However, this child can be a Flex container which allows you to add additional children.

Inheritance Hierarchy

- *FlexUIMovieClip*
  - FlexContainerMovieClip

Syntax

```
'Declaration
Public Class FlexContainerMovieClip _
Inherits FlexUIMovieClip
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexUIMovieClip*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexUIMovieClip*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexUIMovieClip*)
className	the name of this instance's class, such as "Button". (Inherited from *FlexUIMovieClip*)
currentState	the current state of this component. (Inherited from *FlexUIMovieClip*)
enabled	whether a movie clip is enabled. (Inherited from *FlexUIMovieClip*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexUIMovieClip*)
errorString	the text that is displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexUIMovieClip*)
focusEnabled	whether the component can receive focus when selected. (Inherited from *FlexUIMovieClip*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexUIMovieClip*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexUIMovieClip*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. Valid values are 0 to 100. (Inherited from *FlexUIMovieClip*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Valid values are 0 to 100. (Inherited from *FlexUIMovieClip*)
scaleX	the horizontal scale (percentage) of the object as applied from the registration point. The default registration point is (0,0). 1.0 equals 100% scale. (Inherited from *FlexUIMovieClip*)

Name	Description
scaleY	the vertical scale (percentage) of an object as applied from the registration point of the object. The default registration point is (0,0). 1.0 is 100% scale. (Inherited from *FlexUIMovieClip*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexUIMovieClip*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexUIMovieClip*)
useHandCursor	whether the pointing hand (hand cursor) appears when the mouse rolls over a sprite in which the buttonMode property is set to true. (Inherited from *FlexUIMovieClip*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Defines the value of the type property of a keyFocusChange event object. (Inherited from *FlexUIMovieClip*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Called by the FocusManager when the component receives focus. The component may in turn set focus to an internal component. (Inherited from *FlexUIMovieClip*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexDataGrid Class

Description

The DataGrid control is like a List except that it can show more than one column of data making it suited for showing objects with multiple properties.

Inheritance Hierarchy

- *FlexListBase*
  - FlexDataGrid

Syntax

```
'Declaration
Public Class FlexDataGrid _
Inherits FlexListBase
```

Properties

Name	Description
allowMultipleSelection	whether you can allow more than one item to be selected at the same time. (Inherited from *FlexListBase*)
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alternatingItemColors	the colors to use for the backgrounds of the items in the list. (Inherited from *FlexListBase*)

Name	Description
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundDisabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3. (Inherited from *FlexListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnCount	the number of columns to be displayed in a TileList control or items in a HorizontalList control. (Inherited from *FlexListBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexListBase*)
columnWidth	the width of the control's columns. (Inherited from *FlexListBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipField	the name of the field in the data provider items to display as the data tip. (Inherited from *FlexListBase*)
editable	whether the user can edit items in the data provider.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexListBase*)
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4. (Inherited from *FlexListBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
headerColors	An array of two colors used to draw the header background gradient.
headerStyleName	the name of a CSS style declaration for controlling other aspects of the appearance of the column headers. The default value is *dataGridStyles*.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalGridLineColor	the color of the horizontal grid lines.
horizontalGridLines	whether to show horizontal grid lines between the rows.

Name	Description
iconField	the name of the field in the data provider object that determines what to display as the icon. (Inherited from *FlexListBase*)
id	the ID of the component. (Inherited from *FlexObject*)
labelField	the name of the field in the data provider items to display as the label. (Inherited from *FlexListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexListBase*)
lockedColumnCount	the index of the first column in the control that scrolls. (Inherited from *FlexListBase*)
lockedRowCount	the index of the first row in the control that scrolls. (Inherited from *FlexListBase*)
minColumnWidth	The minimum width of the columns, in pixels. If not NaN, the DataGrid control applies this value as the minimum width for all columns. Otherwise, individual columns can have their own minimum widths. The default value is NaN.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexListBase*)
numRows	the total number of rows of data available. (Inherited from *FlexListBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizableColumns	whether the user can change the size of the columns.
rollOverColor	the color of the background of a renderer when the user rolls over it. The default value is 0xEEFEE6. (Inherited from *FlexListBase*)
rowCount	the number of rows to be displayed. (Inherited from *FlexListBase*)
rowHeight	the height of the rows in pixels. (Inherited from *FlexListBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)

Name	Description
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	whether the list shows selected items as selected. (Inherited from *FlexListBase*)
selectedIndex	the index in the data provider of the selected item. (Inherited from *FlexListBase*)
selectedIndices	an array of indices in the data provider of the selected items. (Inherited from *FlexListBase*)
selectedItem	a reference to the selected item in the data provider. (Inherited from *FlexListBase*)
selectedItems	an array of references to the selected items in the data provider. (Inherited from *FlexListBase*)
selectionColor	the color of the background of a renderer when the user selects it. The default value is 0x7FCEFF. (Inherited from *FlexListBase*)
selectionDisabledColor	the color of the background of a renderer when the component is disabled. The default value is 0xDDDDDD. (Inherited from *FlexListBase*)
showDataTips	whether dataTips are displayed for text in the rows. (Inherited from *FlexListBase*)
sortableColumns	whether the user can sort the data provider items by clicking on a column header cell.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C. (Inherited from *FlexListBase*)
textSelectedColor	the color of the text of a renderer when the user selects it. The default value is 0x2B333C. (Inherited from *FlexListBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useRollOver	whether items are highlighted as the mouse rolls over them. (Inherited from *FlexListBase*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
variableRowHeight	whether the individual rows can have different height. (Inherited from *FlexListBase*)
verticalAlign	the vertical alignment of a renderer in a row. (Inherited from *FlexListBase*)
verticalGridLineColor	the color of the vertical grid lines. The default value is 0x666666.
verticalGridLines	whether to show vertical grid lines between the columns.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
wordWrap	whether text in the row should be word wrapped. (Inherited from *FlexListBase*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
ColumnStretch	Dispatched when a user changes the width of a column, indicating that the amount of data displayed in that column may have changed.
Deselect	Defines the value of the type property of the event object for an event that is dispatched when a previously selected item is deselected. (Inherited from *FlexListBase*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Defines the value of the type property of the ListEvent object for an itemDoubleClick event, which indicates that the user double-clicked

Name	Description
	the mouse over a visual item in the control. (Inherited from *FlexListBase*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexListBase*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexListBase*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexListBase*)
Edit	Defines the value of the type property of the event object for a itemEditBegin event, which indicates that an item is ready to be edited.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexListBase*)

Name	Description
HeaderClick	Dispatched when the user releases the mouse button on a column header to request the control to sort the grid contents based on the contents of the column.
HeaderShift	Dispatched when the user releases the mouse button on a column header after having dragged the column to a new location resulting in shifting the column to a new index.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexScrollBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
MultiSelect	Defines the value of the type property of the event object for an event that is dispatched when an item is selected as part of an action that selects multiple items. (Inherited from *FlexListBase*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event. (Inherited from *FlexListBase*)

Name	Description
ScrollToIndex	Ensures that the data provider item at the given index is visible. (Inherited from *FlexListBase*)
Select	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)
SelectIndex	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object,

Name	Description
	for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexDateChooser Class

Description

Displays the name of a month, the year, and a grid of the days of the month, with columns labeled for the day of the week.

The user can select a date, a range of dates, or multiple dates. The control contains forward and back arrow buttons for changing the month and year.

You can let users select multiple dates, disable the selection of certain dates, and limit the display to a range of dates.

Inheritance Hierarchy

- *FlexObject*
  - FlexDateChooser

Syntax

```
'Declaration
Public Class FlexDateChooser _
Inherits FlexObject
```

Properties

Name	Description
allowDisjointSelection	whether non-contiguous selection is allowed in the DateChooser control.
allowMultipleSelection	whether multiple selection is allowed in the DateChooser control.
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)

Name	Description
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. The default value is 0x0B333C.
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3.
disabledDays	the days to disable in a week.
disabledRanges	the single and multiple days to disable.
displayedMonth	the month displayed in the DateChooser control.
displayedYear	the year displayed in the DateChooser control. The default value is the current year.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillColors	the colors used to tint the background of the control.
firstDayOfWeek	the number representing the day of the week to display in the first column of the DateChooser control.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels.
fontStyle	whether the text is italic font.
fontWeight	whether the text is boldface.
headerColors	the colors of the band at the top of the DateChooser control. The default value is [0xE6EEEE, 0xFFFFFF].
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
maxYear	the last year selectable in the control.
minYear	the first year selectable in the control.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that

Name	Description
	is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
rollOverColor	the color of the highlight area of the date when the user holds the mouse pointer over a date in the DateChooser control. The default value is 0xE3FFD6.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedDate	the date as selected in the DateChooser control.
selectedRanges	the selected date ranges.
selectionColor	the color of the highlight area of the currently selected date in the DateChooser control. The default value is 0xCDFFC1.
showToday	whether today's date is highlighted in the DataChooser control.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	whether the text is boldface.
textDecoration	whether the text is underlined.
textIndent	the offset of first line of text from the left side of the container, in pixels. The default value is 0.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
todayColor	the color of the highlight of today's date in the DateChooser control. The default value is 0x2B333.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
yearNavigationEnabled	whether year navigation is enabled.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when a date is selected or changed.
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollDate	The DateChooserEvent.SCROLL constant defines the value of the type property of the event object for a scrollevent.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object,

Name	Description
	for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexDateField Class

Description

A text field that shows the date with a calendar icon on its right side.

When the user clicks anywhere inside the bounding box of the control, a DateChooser control pops up and shows the dates in the month of the current date. If no date is selected, the text field is blank and the month of the current date is displayed in the DateChooser control.

Inheritance Hierarchy

- *FlexComboBase*
  - FlexDateField

Syntax

```
'Declaration
Public Class FlexDateField _
Inherits FlexComboBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. (Inherited from *FlexComboBase*)
backgroundColor	the background color of a component. (Inherited from *FlexComboBase*)

Name	Description
backgroundImage	the background image of a component. (Inherited from *FlexComboBase*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexComboBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. (Inherited from *FlexComboBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the background color of the component when it is disabled. (Inherited from *FlexComboBase*)
disabledDays	the days to disable in a week.
disabledRanges	the disabled single and multiple days.
displayedMonth	the month that is displayed in the DateChooser control.
displayedYear	the year that is displayed in the DateChooser control.
dropShadowEnabled	whether the ComboBase container's drop shadow is visible. The default value is true. (Inherited from *FlexComboBase*)
editable	whether the control is editable, which lets the user directly type entries that are not specified in the dataProvider, or not editable, which requires the user select from the items in the dataProvider. (Inherited from *FlexComboBase*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alphas used for the background fill of controls. (Inherited from *FlexComboBase*)
fillColors	the colors used to tint the background of the control. (Inherited from *FlexComboBase*)
firstDayOfWeek	the day of the week (0-6, where 0 is the first element of the dayNames Array) to display in the first column of the DateChooser control.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexComboBase*)
fontSize	the height of the text, in pixels. (Inherited from *FlexComboBase*)
fontStyle	whether the text is italic font. (Inherited from *FlexComboBase*)
fontWeight	whether the text is boldface. (Inherited from *FlexComboBase*)

Name	Description
headerColors	the colors of the band at the top of the DateChooser control. The default value is [0xE6EEEE, 0xFFFFFF].
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
highlightColor	the highlight color of the drop-down list. (Inherited from *FlexComboBase*)
id	the ID of the component. (Inherited from *FlexObject*)
maxYear	the last year selectable in the control.
minYear	the first year selectable in the control.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
restrict	the set of characters that a user can or cannot enter into the text field. (Inherited from *FlexComboBase*)
rollOverColor	the color of the highlight area of the date when the user holds the mouse pointer over a date in the DateChooser control. The default value is 0xE3FFD6.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedDate	the date as selected in the DateChooser control.
selectedIndex	the index in the data provider of the selected item. (Inherited from *FlexComboBase*)
showToday	the day highlighted in the DateChooser control.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)

Name	Description
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	the contents of the text field. (Inherited from *FlexComboBase*)
textAlign	the alignment of text within a container. Possible values are:left, right, or center. (Inherited from *FlexComboBase*)
textDecoration	whether the text is underlined. (Inherited from *FlexComboBase*)
textIndent	the offset of first line of text from the left side of the container, in pixels. The default value is 0. (Inherited from *FlexComboBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
todayColor	the color of the highlight of today's date in the DateChooser control. The default value is 0x2B333.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
yearNavigationEnabled	whether the year up and down buttons are available for navigation.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Change	Dispatched when a date is selected or changed, and the DateChooser control closes.
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Defines the value of the type property of the event object for a close event. (Inherited from *FlexComboBase*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Creates an Event object that contains information about text events. Event objects are passed as parameters to event listeners. (Inherited from *FlexComboBase*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
Open	Defines the value of the type property of the event object for a open event. (Inherited from *FlexComboBase*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event. (Inherited from *FlexComboBase*)
ScrollDate	The DateChooserEvent.SCROLL constant defines the value of the type property of the event object for a scrollevent.
SelectText	Lets you track selection within a text field. (Inherited from *FlexComboBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexComboBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexDisplayObject Class

Description
The base class for all objects that can be placed on the display list. The display list manages all objects displayed in Flash Player or Adobe AIR.

Inheritance Hierarchy

- *TestObject*
  - FlexDisplayObject
    - *FlexListLabel*
    - *FlexObject*
    - *FlexUIMovieClip*

Syntax

```
'Declaration
Public Class FlexDisplayObject _
Inherits TestObject _
Implements IClickable, IKeyable
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified.
height	the height of the display object, in pixels.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren.
tabChildren	whether the children of the display object are tab enabled. The default is true.
tabEnabled	whether this object is in the tab order.
tabIndex	the tab ordering of objects in a SWF file.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible.
width	the width of the display object, in pixels.
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer.
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout

Name	Description
	parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexDividedBox Class

Description

Measures and lays out its children horizontally or vertically in exactly the same way as a Box container, but it inserts draggable dividers in the gaps between the children. Users can drag any divider to resize the children on each side.

The *DividedBox* class is the base class for the *HDividedBox* and *VDividedBox* classes.

Inheritance Hierarchy

- *FlexBox*
  - FlexDividedBox

Syntax

```
'Declaration
Public Class FlexDividedBox _
Inherits FlexBox
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)

Name	Description
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
direction	whether to use vertical (default) or horizontal layout. (Inherited from *FlexBox*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)

Name	Description
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAlign	the horizontal alignment of children in the container. (Inherited from *FlexBox*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizeToContent	whether the DividedBox automatically resizes to the size of its children.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAlign	the vertical alignment of children in the container. (Inherited from *FlexBox*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)

Name	Description
Dragged	Dispatched multiple times as the user drags any divider. The dividerDrag event is dispatched after the dividerPress event and before the dividerRelease event.
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
Pressed	Dispatched when the user presses any divider in this container.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Released	Dispatched when the user releases a divider.
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexForm Class

Description
Use this class to control the layout of a form, mark form fields as required or optional, handle error messages, and bind your form data to the Flex data model to perform data checking and validation. This class also lets you use style sheets to configure the appearance of your forms.

Inheritance Hierarchy

- *FlexContainer*
  - FlexForm

Syntax

```
'Declaration
Public Class FlexForm _
Inherits FlexContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)

Name	Description
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)

Name	Description
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexFormItem Class

Description
Defines a label and one or more children arranged horizontally or vertically. Children can be controls or other containers. A single Form container can hold multiple FormItem containers.

Inheritance Hierarchy

- *FlexContainer*
  - FlexFormItem

Syntax

```
'Declaration
Public Class FlexFormItem _
Inherits FlexContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAlign	the horizontal alignment of children in the container.
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)

Name	Description
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)

Name	Description
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexHLOCChart Class

Description

The HLOCChart (High Low Open Close) control represents financial data as a series of elements representing the high, low, closing, and optionally opening values of a data series.

The top and bottom of the vertical line in each element represent the high and low values for the datapoint. The right-facing tick represents the closing values, and the left tick represents the opening value if one was specified.

Inheritance Hierarchy

- *FlexCartesianChart*
  - FlexHLOCChart

Syntax

```
'Declaration
Public Class FlexHLOCChart _
Inherits FlexCartesianChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)
columnWidthRatio	a ratio of how wide to draw the HLOC lines relative to the horizontal axis's category widths, as a percentage in the range of 0 to 1.
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)
description	a short description of the data in the chart. (Inherited from *FlexChart*)
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)

Name	Description
gutterBottom	The size of the region, in pixels, between the bottom of the chart data area and the bottom of the chart control. (Inherited from *FlexCartesianChart*)
gutterLeft	The size of the region, in pixels, between the left of the chart data area and the left of the chart control. (Inherited from *FlexCartesianChart*)
gutterRight	The size of the region, in pixels, between the right side of the chart data area and the outside of the chart control. (Inherited from *FlexCartesianChart*)
gutterTop	The size of the region, in pixels, between the top of the chart data area and the top of the chart control. (Inherited from *FlexCartesianChart*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAxisRatio	the height limit of the horiztonal axis. (Inherited from *FlexCartesianChart*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maxColumnWidth	how wide to draw the HLOC lines, in pixels.
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)

Name	Description
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexCartesianChart*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
verticalAxisRatio	the width limit of the vertical axis. (Inherited from *FlexCartesianChart*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexHLOCSeries Class

Description

Represents financial data as a series of elements representing the high, low, closing, and, optionally, opening values of a data series.

The top and bottom of the vertical line in each element represent the high and low values for the datapoint. The right-facing tick mark represents the closing value, and the left tick mark represents the opening value, if one was specified.

Inheritance Hierarchy

- *FlexHLOCSeriesBase*
  - FlexHLOCSeries

Syntax

```
'Declaration
Public Class FlexHLOCSeries _
Inherits FlexHLOCSeriesBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)

Name	Description
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
columnWidthRatio	the width of elements relative to the category width. (Inherited from *FlexHLOCSeriesBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
maxColumnWidth	the width of the elements, in pixels. (Inherited from *FlexHLOCSeriesBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)

Name	Description
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
xField	the field of the data provider that determines the x-axis location of the element. (Inherited from *FlexHLOCSeriesBase*)

Name	Description
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)

Name	Description
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexHLOCSeriesBase Class

Description

Represents financial data as a series of elements representing the high, low, closing, and, optionally, opening values of a data series.

The top and bottom of the vertical line in each element represent the high and low values for the datapoint. The right-facing tick mark represents the closing value, and the left tick mark represents the opening value, if one was specified.

Inheritance Hierarchy

- *FlexChartSeries*
  - FlexHLOCSeriesBase
    - *FlexCandlestickSeries*
    - *FlexHLOCSeries*

Syntax

```
'Declaration
Public Class FlexHLOCSeriesBase _
Inherits FlexChartSeries
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
columnWidthRatio	the width of elements relative to the category width.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)

Name	Description
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
maxColumnWidth	the width of the elements, in pixels.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)

Name	Description
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
xField	the field of the data provider that determines the x-axis location of the element.
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexImage Class

Description

Lets you import JPEG, PNG, GIF, and SWF files at runtime.

You can also embed any of these files and SVG files at compile time by using @Embed(source='filename').

Note: Flex also includes the SWFLoader control for loading Flex applications. You typically use the Image control for loading static graphic files and SWF files, and use the SWFLoader control for loading Flex applications. The Image control is also designed to be used in custom item renderers and item editors.

Inheritance Hierarchy

- *FlexLoader*
  - `FlexImage`

Syntax

```
'Declaration
Public Class FlexImage _
Inherits FlexLoader
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is

Name	Description
	NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
source	The URL, object, class or string name of a class to load as the content. (Inherited from *FlexLoader*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexLabel Class

Description
Displays a single line of noneditable text.

Inheritance Hierarchy

- *FlexObject*
  - FlexLabel

Syntax
```
'Declaration
Public Class FlexLabel _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)

Name	Description
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. The default value is 0x0B333C.
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels.
fontStyle	whether the text is italic font.
fontWeight	whether the text is boldface.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
htmlText	the text displayed by the Label control, including HTML markup that expresses the styles of that text.
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)

Name	Description
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	the plain text displayed by this control. Its appearance is determined by the CSS styles of this Label control.
textAlign	the alignment of text within a container.
textDecoration	whether the text is underlined.
textIndent	the offset of the first line of text from the left side of the container, in pixels. The default value is 0.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexLineChart Class

Description

Represents a data series as points connected by a continuous line.

You can use an icon or symbol to represent each data point along the line, or show a simple line with no icons.

Inheritance Hierarchy

- *FlexCartesianChart*
  - FlexLineChart

Syntax

```
'Declaration
Public Class FlexLineChart _
Inherits FlexCartesianChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)
description	a short description of the data in the chart. (Inherited from *FlexChart*)
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)

Name	Description
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)
gutterBottom	The size of the region, in pixels, between the bottom of the chart data area and the bottom of the chart control. (Inherited from *FlexCartesianChart*)
gutterLeft	The size of the region, in pixels, between the left of the chart data area and the left of the chart control. (Inherited from *FlexCartesianChart*)
gutterRight	The size of the region, in pixels, between the right side of the chart data area and the outside of the chart control. (Inherited from *FlexCartesianChart*)
gutterTop	The size of the region, in pixels, between the top of the chart data area and the top of the chart control. (Inherited from *FlexCartesianChart*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAxisRatio	the height limit of the horiztonal axis. (Inherited from *FlexCartesianChart*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)

Name	Description
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexCartesianChart*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAxisRatio	the width limit of the vertical axis. (Inherited from *FlexCartesianChart*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexLineSeries Class

Description
Defines a data series for a LineChart control.

Inheritance Hierarchy

- *FlexChartSeries*
  - FlexLineSeries

Syntax

```
'Declaration
Public Class FlexLineSeries _
Inherits FlexChartSeries
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)

Name	Description
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
form	the line type for the chart.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is

Name	Description
	NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
radius	the radius, in pixels, of the chart elements for the data points.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
sortOnXField	whether the line datapoints are sorted from left to right before rendering.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
xField	the field of the data provider that determines the x-axis location of each data point.

Name	Description
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
yField	the field of the data provider that determines the y-axis location of each data point.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)

Name	Description
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexLinkBar Class

Description

Defines a horizontal or vertical row of LinkButton controls that designate a series of link destinations.

You typically use a LinkBar control to control the active child container of a ViewStack container, or to create a stand-alone set of links.

Inheritance Hierarchy

- *FlexNavigationBar*
  - FlexLinkBar

Syntax

```
'Declaration
Public Class FlexLinkBar _
Inherits FlexNavigationBar
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
direction	whether to use vertical (default) or horizontal layout. (Inherited from *FlexBox*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAlign	the horizontal alignment of children in the container. (Inherited from *FlexBox*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list,

Name	Description
	use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the active navigation item, where the first item is at an index of 0. The default value is -1. (Inherited from *FlexNavigationBar*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAlign	the vertical alignment of children in the container. (Inherited from *FlexBox*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)

Name	Description
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Defines the value of the type property of the event object for an itemClick event. (Inherited from *FlexNavigationBar*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexNavigationBar*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexList Class

Description
Displays a list of items horizontally, verically, or laid out in tiles.

Inheritance Hierarchy

- *FlexListBase*
  - FlexList
    - *FlexTree*

Syntax
```
'Declaration
Public Class FlexList _
Inherits FlexListBase
```

Properties

Name	Description
allowMultipleSelection	whether you can allow more than one item to be selected at the same time. (Inherited from *FlexListBase*)
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alternatingItemColors	the colors to use for the backgrounds of the items in the list. (Inherited from *FlexListBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)

Name	Description
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundDisabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3. (Inherited from *FlexListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnCount	the number of columns to be displayed in a TileList control or items in a HorizontalList control. (Inherited from *FlexListBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexListBase*)
columnWidth	the width of the control's columns. (Inherited from *FlexListBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipField	the name of the field in the data provider items to display as the data tip. (Inherited from *FlexListBase*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexListBase*)
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4. (Inherited from *FlexListBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
iconField	the name of the field in the data provider object that determines what to display as the icon. (Inherited from *FlexListBase*)
id	the ID of the component. (Inherited from *FlexObject*)
labelField	the name of the field in the data provider items to display as the label. (Inherited from *FlexListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexListBase*)
lockedColumnCount	the index of the first column in the control that scrolls. (Inherited from *FlexListBase*)
lockedRowCount	the index of the first row in the control that scrolls. (Inherited from *FlexListBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to

Name	Description
	false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexListBase*)
numRows	the total number of rows of data available. (Inherited from *FlexListBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
rollOverColor	the color of the background of a renderer when the user rolls over it. The default value is 0xEEFEE6. (Inherited from *FlexListBase*)
rowCount	the number of rows to be displayed. (Inherited from *FlexListBase*)
rowHeight	the height of the rows in pixels. (Inherited from *FlexListBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	whether the list shows selected items as selected. (Inherited from *FlexListBase*)
selectedIndex	the index in the data provider of the selected item. (Inherited from *FlexListBase*)
selectedIndices	an array of indices in the data provider of the selected items. (Inherited from *FlexListBase*)
selectedItem	a reference to the selected item in the data provider. (Inherited from *FlexListBase*)
selectedItems	an array of references to the selected items in the data provider. (Inherited from *FlexListBase*)
selectionColor	the color of the background of a renderer when the user selects it. The default value is 0x7FCEFF. (Inherited from *FlexListBase*)
selectionDisabledColor	the color of the background of a renderer when the component is disabled. The default value is 0xDDDDDD. (Inherited from *FlexListBase*)
showDataTips	whether dataTips are displayed for text in the rows. (Inherited from *FlexListBase*)

Name	Description
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C. (Inherited from *FlexListBase*)
textSelectedColor	the color of the text of a renderer when the user selects it. The default value is 0x2B333C. (Inherited from *FlexListBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useRollOver	whether items are highlighted as the mouse rolls over them. (Inherited from *FlexListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
variableRowHeight	whether the individual rows can have different height. (Inherited from *FlexListBase*)
verticalAlign	the vertical alignment of a renderer in a row. (Inherited from *FlexListBase*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
wordWrap	whether text in the row should be word wrapped. (Inherited from *FlexListBase*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Deselect	Defines the value of the type property of the event object for an event that is dispatched when a previously selected item is deselected. (Inherited from *FlexListBase*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Defines the value of the type property of the ListEvent object for an itemDoubleClick event, which indicates that the user double-clicked the mouse over a visual item in the control. (Inherited from *FlexListBase*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexListBase*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexListBase*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexListBase*)
Edit	Defines the value of the type property of the event object for a itemEditBegin event, which indicates that an item is ready to be edited.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexListBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexScrollBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
MultiSelect	Defines the value of the type property of the event object for an event that is dispatched when an item is selected as part of an action that selects multiple items. (Inherited from *FlexListBase*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event. (Inherited from *FlexListBase*)
ScrollToIndex	Ensures that the data provider item at the given index is visible. (Inherited from *FlexListBase*)
Select	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)
SelectIndex	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexListBase Class

Description

The ListBase class is the base class for controls that represent lists of items that can have one or more selected and can scroll through the items.

Items are supplied using the dataProvider property and displayed via item renderers.

In a model/view architecture, the ListBase-derived class represents the view, and the dataProvider object represents the model.

Inheritance Hierarchy

- *FlexScrollBase*
  - FlexListBase
    - *FlexAdvancedDataGrid*
    - *FlexDataGrid*
    - *FlexList*

Syntax

```
'Declaration
Public Class FlexListBase _
Inherits FlexScrollBase
```

Properties

Name	Description
allowMultipleSelection	whether you can allow more than one item to be selected at the same time.
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alternatingItemColors	the colors to use for the backgrounds of the items in the list.
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundDisabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnCount	the number of columns to be displayed in a TileList control or items in a HorizontalList control.
columnNames	a list containing the names of all columns in the data.
columnWidth	the width of the control's columns.
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipField	the name of the field in the data provider items to display as the data tip.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
iconField	the name of the field in the data provider object that determines what to display as the icon.
id	the ID of the component. (Inherited from *FlexObject*)

Name	Description
labelField	the name of the field in the data provider items to display as the label.
lastVisibleRow	the index of the last visible child.
lockedColumnCount	the index of the first column in the control that scrolls.
lockedRowCount	the index of the first row in the control that scrolls.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
rollOverColor	the color of the background of a renderer when the user rolls over it. The default value is 0xEEFEE6.
rowCount	the number of rows to be displayed.
rowHeight	the height of the rows in pixels.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	whether the list shows selected items as selected.
selectedIndex	the index in the data provider of the selected item.
selectedIndices	an array of indices in the data provider of the selected items.
selectedItem	a reference to the selected item in the data provider.
selectedItems	an array of references to the selected items in the data provider.
selectionColor	the color of the background of a renderer when the user selects it. The default value is 0x7FCEFF.
selectionDisabledColor	the color of the background of a renderer when the component is disabled. The default value is 0xDDDDDD.
showDataTips	whether dataTips are displayed for text in the rows.

Name	Description
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C.
textSelectedColor	the color of the text of a renderer when the user selects it. The default value is 0x2B333C.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useRollOver	whether items are highlighted as the mouse rolls over them.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
variableRowHeight	whether the individual rows can have different height.
verticalAlign	the vertical alignment of a renderer in a row.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
wordWrap	whether text in the row should be word wrapped.
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Deselect	Defines the value of the type property of the event object for an event that is dispatched when a previously selected item is deselected.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Defines the value of the type property of the ListEvent object for an itemDoubleClick event, which indicates that the user double-clicked the mouse over a visual item in the control.
DragCancel	Defines the value of the type property of the event object for a dragComplete event.
DragDrop	Defines the value of the type property of the event object for a dragDrop event.
DragStart	Defines the value of the type property of the event object for a dragStart event.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexScrollBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
MultiSelect	Defines the value of the type property of the event object for an event that is dispatched when an item is selected as part of an action that selects multiple items.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Scroll	Defines the value of the type property of the event object for a scroll event.
ScrollToIndex	Ensures that the data provider item at the given index is visible.
Select	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected.
SelectIndex	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any

Name	Description
	results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexListLabel Class

Description
Defines the default item renderer for a control. By default, the item renderer draws the text associated with each item in the list, and an optional icon.

Inheritance Hierarchy

- *FlexDisplayObject*
  - FlexListLabel

Syntax

```
'Declaration
Public Class FlexListLabel _
Inherits FlexDisplayObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element.
automationIndex	a string containing the automation index for the element.
automationName	the name that can be used as an identifier for this object.
automationValue	the value that generally corresponds to the rendered appearance of the object and should be usable for correlating the identifier with the object as it appears visually within the application.
className	the name of this instance's class, such as "Button".
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the

Name	Description
	mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexLoader Class

Description
Loads and displays a specified SWF file. You typically use SWFLoader for loading one Flex application into a host Flex application.

Inheritance Hierarchy

- *FlexObject*
  - FlexLoader
    - *FlexImage*

Syntax

```
'Declaration
Public Class FlexLoader _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
source	The URL, object, class or string name of a class to load as the content.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexMenu Class

Description
Creates a pop-up menu of individually selectable choices, similar to the File or Edit menu found in most software applications. The popped up menu can have as many levels of submenus as needed.

Inheritance Hierarchy

- *FlexObject*
  - FlexMenu

Syntax

```
'Declaration
Public Class FlexMenu _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alternatingItemColors	the colors used for menu or submenu menu items in an alternating pattern.
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
rollOverColor	the color of the menu item background when a user rolls the mouse over it. The default value is 0xB2E1FF.
rowCount	the number of rows to be displayed.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index in the data provider of the selected item.
selectionColor	the color of the menu item background when a menu item is selected. The default value is 0x7FCEFF.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
textRollOverColor	the color of the menu item text when a user rolls the mouse over the menu item. The default value is 0x2B333C.
textSelectedColor	the color of the menu item text when the menu item is selected. The default value is 0x2B333C.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
Hide	Dispatched when a menu or submenu is dismissed.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Dispatched when a menu item is selected.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Show	Dispatched when a menu or submenu opens.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexMenuBar Class

Description
Defines a horizontal, top-level menu bar that contains one or more menus. Clicking on a top-level menu item opens a pop-up submenu that is an instance of the Menu control.

Inheritance Hierarchy

- *FlexObject*
  - FlexMenuBar

Syntax
```
'Declaration
Public Class FlexMenuBar _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)

Name	Description
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundColor	the background color of the component.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. The default value is 0x0B333C.
columnNames	a list containing the names of all columns in the data.
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillColors	the colors used to tint the background of the control.
firstVisibleRow	the index of the first visible child.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels.
fontStyle	whether the text is italic font.
fontWeight	whether the text is boldface.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)

Name	Description
numChildren	the number of children of this object.
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
rollOverColor	the color of the menu item background when a user rolls the mouse over it.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index in the MenuBar control of the currently open Menu or the last opened Menu if none are currently open.
selectionColor	the color of the menu item background when a menu item is selected.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container.
textDecoration	whether the text is underlined.
textIndent	the offset of the first line of text from the left side of the container, in pixels. The default value is 0.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)

Name	Description
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
Hide	Dispatched when a menu or submenu is dismissed.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Show	Dispatched when a menu or submenu opens.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexNavigationBar Class

Description
The superclass for navigator controls, such as the LinkBar and TabBar controls. This class cannot be instantiated directly.

Inheritance Hierarchy

- *FlexBox*
  - FlexNavigationBar
    - *FlexButtonBar*
    - *FlexLinkBar*

Syntax

```
'Declaration
Public Class FlexNavigationBar _
Inherits FlexBox
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
direction	whether to use vertical (default) or horizontal layout. (Inherited from *FlexBox*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAlign	the horizontal alignment of children in the container. (Inherited from *FlexBox*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list,

Name	Description
	use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the active navigation item, where the first item is at an index of 0. The default value is -1.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAlign	the vertical alignment of children in the container. (Inherited from *FlexBox*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)

Name	Description
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Defines the value of the type property of the event object for an itemClick event.
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown.

Name	Description
	The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexNumericStepper Class

Description

Lets the user select a number from an ordered set.

The NumericStepper control consists of a single-line input text field and a pair of arrow buttons for stepping through the possible values. The Up Arrow and Down Arrow keys also cycle through the values.

Inheritance Hierarchy

- *FlexObject*
  - FlexNumericStepper

Syntax

```
'Declaration
Public Class FlexNumericStepper _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
color	the color of text in the component, including the component label. The default value is 0x0B333C.
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillColors	the colors used to tint the background of the control. Pass the same color for both values for a flat-looking control. The default value is [0xFFFFFF, 0xCCCCCC].
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels.
fontStyle	whether the text is italic font.
fontWeight	whether the text is boldface. Recognized values are normal and bold.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
maximum	the maximum value of the NumericStepper.
minimum	the minimum value of the NumericStepper.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
nextValue	the value that is one step larger than the current value property and not greater than the maximum property value.
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is

Name	Description
	NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
previousValue	the value that is one step smaller than the current value property and not smaller than the maximum property value.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
stepSize	the non-zero unit of change between values.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. Possible values are: left, right, or center.
textDecoration	whether the text is underlined. Possible values are *none* and *underline*. The default value is *none*.
textIndent	the offset of first line of text from the left side of the container, in pixels. The default value is 0.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	the value displayed in the text area of the NumericStepper control.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when the value of the NumericStepper control changes as a result of user interaction.
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Creates an Event object that contains information about text events. Event objects are passed as parameters to event listeners.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SelectText	Lets you track selection within a text field.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexObject Class

Description
The base class for all visual components, both interactive and noninteractive.

Inheritance Hierarchy

- *FlexDisplayObject*
  - FlexObject
    - *FlexAxisRenderer*
    - *FlexButton*
    - *FlexChart*
    - *FlexChartLegend*
    - *FlexChartSeries*
    - *FlexComboBase*
    - *FlexContainer*
    - *FlexDateChooser*
    - *FlexLabel*
    - *FlexLoader*
    - *FlexMenu*
    - *FlexMenuBar*
    - *FlexNumericStepper*
    - *FlexProgressBar*
    - *FlexRule*
    - *FlexScrollBar*
    - *FlexScrollBase*
    - *FlexSlider*
    - *FlexVideoDisplay*
    - *SparkDataGridLabel*
    - *SparkGroupBase*
    - *SparkObject*
    - *SparkPopUpAnchor*
    - *SparkTextBase*
    - *SparkVideoDisplay*

Syntax

```
'Declaration
Public Class FlexObject _
Inherits FlexDisplayObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element.
automationIndex	a string containing the automation index for the element.
automationName	the name that can be used as an identifier for this object.
className	the name of this instance's class, such as *Button*.
currentState	the current view state of the component.
enabled	whether the component can accept user interaction.

Name	Description
errorColor	the color of the component highlight when validation fails.
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails.
focusEnabled	whether the component can receive focus when tabbed to.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size.
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN.
scaleX	the number that specifies the horizontal scaling factor.
scaleY	the number that specifies the vertical scaling factor.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component.
toolTip	the text to display in the ToolTip. The default value is `Nothing`.
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)

Name	Description
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexOLAPDataGrid Class

Description

Expands on the functionality of the AdvancedDataGrid control to add support for the display of the results of OLAP queries.

Like all Flex data grid controls, the OLAPDataGrid control is designed to display data in a two-dimensional representation of rows and columns.

Inheritance Hierarchy

- *FlexAdvancedDataGrid*
  - FlexOLAPDataGrid

Syntax

```
'Declaration
Public Class FlexOLAPDataGrid _
Inherits FlexAdvancedDataGrid
```

Properties

Name	Description
allowDragSelection	whether drag-selection is enabled. Drag-Selection is the ability to select an item by dragging into it as opposed to normal selection where you can't have the mouse button down when you mouse over the item you want to select. (Inherited from *FlexAdvancedDataGrid*)
allowMultipleSelection	whether you can allow more than one item to be selected at the same time. (Inherited from *FlexListBase*)
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alternatingItemColors	the colors to use for the backgrounds of the items in the list. (Inherited from *FlexListBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundDisabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3. (Inherited from *FlexListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnCount	the number of columns to be displayed in a TileList control or items in a HorizontalList control. (Inherited from *FlexListBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexListBase*)

Name	Description
columnWidth	the width of the control's columns. (Inherited from *FlexListBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipField	the name of the field in the data provider items to display as the data tip. (Inherited from *FlexListBase*)
defaultCellString	the String displayed in a cell when the data for that cell returned by the IOLAPResult instance is `Nothing` or NaN.
depthColors	the array of colors used for the rows of each level of the navigation tree of the AdvancedDataGrid control, in descending order. The default value is undefined. (Inherited from *FlexAdvancedDataGrid*)
disabledColor	the color of text in the component if it is disabled. (Inherited from *FlexAdvancedDataGrid*)
displayItemsExpanded	whether the navigation tree is expanded to show all items. (Inherited from *FlexAdvancedDataGrid*)
editable	whether the user can edit items in the data provider. (Inherited from *FlexAdvancedDataGrid*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexListBase*)
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4. (Inherited from *FlexListBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
headerColors	an array of two colors used to draw the header background gradient. (Inherited from *FlexAdvancedDataGrid*)
headerHeight	the height of the header cell of the column, in pixels. (Inherited from *FlexAdvancedDataGrid*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalGridLineColor	the color of the horizontal grid lines. (Inherited from *FlexAdvancedDataGrid*)
iconField	the name of the field in the data provider object that determines what to display as the icon. (Inherited from *FlexListBase*)
id	the ID of the component. (Inherited from *FlexObject*)
indentation	the indentation for each node of the navigation tree, in pixels. The default value is 17. (Inherited from *FlexAdvancedDataGrid*)

Name	Description
labelField	the name of the field in the data provider items to display as the label. (Inherited from *FlexListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexListBase*)
lockedColumnCount	the index of the first column in the control that scrolls. (Inherited from *FlexListBase*)
lockedRowCount	the index of the first row in the control that scrolls. (Inherited from *FlexListBase*)
minColumnWidth	the minimum width of the columns, in pixels. (Inherited from *FlexAdvancedDataGrid*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexListBase*)
numRows	the total number of rows of data available. (Inherited from *FlexListBase*)
openDuration	the length of an open or close transition for the navigation tree, in milliseconds. The default value is 250. (Inherited from *FlexAdvancedDataGrid*)
paddingBottom	the number of pixels between the bottom of the row and the bottom of the renderer in the row. (Inherited from *FlexAdvancedDataGrid*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexAdvancedDataGrid*)
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexAdvancedDataGrid*)
paddingTop	the number of pixels between the top of the row and the top of the renderer in the row. (Inherited from *FlexAdvancedDataGrid*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizableColumns	whether the user can change the size of the columns. (Inherited from *FlexAdvancedDataGrid*)

Name	Description
rollOverColor	the color of the background of a renderer when the user rolls over it. The default value is 0xEEFEE6. (Inherited from *FlexListBase*)
rowCount	the number of rows to be displayed. (Inherited from *FlexListBase*)
rowHeight	the height of the rows in pixels. (Inherited from *FlexListBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	whether the list shows selected items as selected. (Inherited from *FlexListBase*)
selectedCells	an Array of cell locations as row and column indices. (Inherited from *FlexAdvancedDataGrid*)
selectedIndex	the index in the data provider of the selected item. (Inherited from *FlexListBase*)
selectedIndices	an array of indices in the data provider of the selected items. (Inherited from *FlexListBase*)
selectedItem	a reference to the selected item in the data provider. (Inherited from *FlexListBase*)
selectedItems	an array of references to the selected items in the data provider. (Inherited from *FlexListBase*)
selectionColor	the color of the background of a renderer when the user selects it. The default value is 0x7FCEFF. (Inherited from *FlexListBase*)
selectionDisabledColor	the color of the background of a renderer when the component is disabled. The default value is 0xDDDDDD. (Inherited from *FlexListBase*)
selectionMode	the selection mode of the control. (Inherited from *FlexAdvancedDataGrid*)
showDataTips	whether dataTips are displayed for text in the rows. (Inherited from *FlexListBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C. (Inherited from *FlexListBase*)
textSelectedColor	the color of the text of a renderer when the user selects it. The default value is 0x2B333C. (Inherited from *FlexListBase*)

Name	Description
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useRollOver	whether items are highlighted as the mouse rolls over them. (Inherited from *FlexListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
variableRowHeight	whether the individual rows can have different height. (Inherited from *FlexListBase*)
verticalAlign	the vertical alignment of a renderer in a row. (Inherited from *FlexListBase*)
verticalGridLineColor	the color of the vertical grid lines. The default value is 0x666666. (Inherited from *FlexAdvancedDataGrid*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
wordWrap	whether text in the row should be word wrapped. (Inherited from *FlexListBase*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
Close	Closes or collapses a AdvancedDataGrid branch. (Inherited from *FlexAdvancedDataGrid*)
ColumnGroupedADGHeaderShift	Dispatched when the user releases the mouse button on a column header after having dragged the column to a new location resulting in shifting the column to a new index. (Inherited from *FlexAdvancedDataGrid*)
ColumnStretch	Dispatched when a user changes the width of a column, indicating that the amount of data displayed in that column may have changed. (Inherited from *FlexAdvancedDataGrid*)
Deselect	Defines the value of the type property of the event object for an event that is dispatched when a previously selected item is deselected. (Inherited from *FlexListBase*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Defines the value of the type property of the ListEvent object for an itemDoubleClick event, which indicates that the user double-clicked the mouse over a visual item in the control. (Inherited from *FlexListBase*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexListBase*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexListBase*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexListBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCellData	Returns data for a cell in the grid. (Inherited from *FlexAdvancedDataGrid*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetGroupedItemChildrenCount	Returns the number of children within the first item of a group. (Inherited from *FlexAdvancedDataGrid*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetRowData	Returns data for the row of the item in the data provider. (Inherited from *FlexAdvancedDataGrid*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexListBase*)
HeaderClick	Dispatched when the user releases the mouse button on a column header to request the control to sort the grid contents based on the contents of the column. (Inherited from *FlexAdvancedDataGrid*)
HeaderShift	Dispatched when the user releases the mouse button on a column header after having dragged the column to a new location resulting in shifting the column to a new index. (Inherited from *FlexAdvancedDataGrid*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsGroupedItem	Returns a value that specifies whether the item is a member of a group. (Inherited from *FlexAdvancedDataGrid*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexScrollBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
MultiSelect	Defines the value of the type property of the event object for an event that is dispatched when an item is selected as part of an action that selects multiple items. (Inherited from *FlexListBase*)
Open	Opens or expands an AdvancedDataGrid branch. (Inherited from *FlexAdvancedDataGrid*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event. (Inherited from *FlexListBase*)
ScrollToIndex	Ensures that the data provider item at the given index is visible. (Inherited from *FlexListBase*)
Select	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)
SelectIndex	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexPanel Class

Description

Consists of a title bar, a caption, a border, and a content area for its children.

Typically, you use Panel containers to wrap top-level application modules. For example, you can include a shopping cart in a Panel container.

Inheritance Hierarchy

- *FlexContainer*
  - FlexPanel
    - *FlexAlert*
    - *FlexTitleWindow*

Syntax

```
'Declaration
Public Class FlexPanel _
Inherits FlexContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dropShadowEnabled	whether the Panel container's drop shadow is visible. The default value is true.

Name	Description
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
footerColors	the array of two colors used to draw the footer (area for the ControlBar container) background.
headerColors	the array of two colors used to draw the header.
headerHeight	the height of the header. The default value is based on the style of the title text.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)

Name	Description
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
status	the text in the status area of the title bar. The default value is "".
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
title	the title or caption displayed in the title bar. The default value is "".
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)

Name	Description
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexPieChart Class

Description

Represents a data series as a standard pie chart.

The data for the data provider determines the size of each wedge in the pie chart relative to the other wedges. You can use the PieSeries class to create standard pie charts, doughnut charts, or stacked pie charts.

Inheritance Hierarchy

- *FlexChart*
  - FlexPieChart

Syntax

```
'Declaration
Public Class FlexPieChart _
Inherits FlexChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)
description	a short description of the data in the chart. (Inherited from *FlexChart*)

Name	Description
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
innerRadius	the size of the hole in the center of the pie chart.
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)

Name	Description
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexPieSeries Class

Description
Defines the data series for a PieChart control.

Inheritance Hierarchy

- *FlexChartSeries*
  - FlexPieSeries

Syntax

```
'Declaration
Public Class FlexPieSeries _
Inherits FlexChartSeries
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
calloutGap	how much space, in pixels, to insert between the edge of the pie and the labels when rendering callouts. The default value is 10.

Name	Description
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
explodeRadius	a number from 0 to 1, specifying how far all wedges of the pie series should be exploded from the center of the chart as a percentage of the total radius.
field	the field of the data provider that determines the data for each wedge of the PieChart control. The default value is `Nothing`.
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontSize	the height of the text, in pixels.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
innerRadius	a number from 0 to 1 specifying the distance from the center of the series to the inner edge of the rendered wedges, as a percentage of the total radius assigned to the series.
insideLabelSizeLimit	the font size threshold, in points, below which inside labels are considered illegible.
labelPosition	how to render value labels.
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
maxLabelRadius	the maximum amount of the PieSeries's radius that can be allocated to labels.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
nameField	the field of the data provider that determines the name of each wedge of the PieChart control. The default value is Nothing.
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)
outerRadius	the percentage of the total space available to the PieSeries to use when rendering the contents of the series.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
startAngle	the starting angle for the first slice of the PieChart control.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is Nothing. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use

Name	Description
	WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexPlotChart Class

Description

Represents data with two values for each data point. One value determines the position of the data point along the horizontal axis, and one value determines its position along the vertical axis.

Inheritance Hierarchy

- *FlexCartesianChart*
  - FlexPlotChart

Syntax

```
'Declaration
Public Class FlexPlotChart _
Inherits FlexCartesianChart
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipContent	whether Flex clips the chart to the area bounded by the axes. Set to false to clip the chart. Set to true to avoid clipping when the data is rendered. (Inherited from *FlexChart*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChart*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipMode	how Flex displays DataTip controls for the chart. (Inherited from *FlexChart*)
description	a short description of the data in the chart. (Inherited from *FlexChart*)
dragEnabled	whether you can drag items out of this chart and drop them on other controls. (Inherited from *FlexChart*)
dropEnabled	whether dragged items can be dropped onto the chart. (Inherited from *FlexChart*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChart*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexChart*)
fontSize	the height of the text, in pixels. The default value is 10 for all controls except the ColorPicker control. For the ColorPicker control, the default value is 11. (Inherited from *FlexChart*)
gutterBottom	The size of the region, in pixels, between the bottom of the chart data area and the bottom of the chart control. (Inherited from *FlexCartesianChart*)
gutterLeft	The size of the region, in pixels, between the left of the chart data area and the left of the chart control. (Inherited from *FlexCartesianChart*)
gutterRight	The size of the region, in pixels, between the right side of the chart data area and the outside of the chart control. (Inherited from *FlexCartesianChart*)
gutterTop	The size of the region, in pixels, between the top of the chart data area and the top of the chart control. (Inherited from *FlexCartesianChart*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAxisRatio	the height limit of the horiztonal axis. (Inherited from *FlexCartesianChart*)
id	the ID of the component. (Inherited from *FlexObject*)

Name	Description
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChart*)
maximumDataTipCount	the maximum number of datatips a chart will show. (Inherited from *FlexChart*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseSensitivity	the distance, in pixels, that Flex considers a data point to be under the mouse pointer when the pointer moves around a chart. (Inherited from *FlexChart*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChart*)
numRows	the total number of rows of data available. (Inherited from *FlexChart*)
paddingBottom	the number of pixels between the chart's bottom border and its content area. (Inherited from *FlexChart*)
paddingLeft	the number of pixels between the control's left border and the left edge of its content area. The default value is 2. (Inherited from *FlexChart*)
paddingRight	the number of pixels between the control's right border and the right edge of its content area. The default value is 0. (Inherited from *FlexChart*)
paddingTop	the number of pixels between the chart's top border and its content area. (Inherited from *FlexChart*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChartItem	the selected ChartItem in the chart. If multiple items are selected, this property specifies the most recently selected item. (Inherited from *FlexChart*)

Name	Description
selectedChartItems	an array of all the selected ChartItem objects in the chart. (Inherited from *FlexChart*)
selectionMode	whether ChartItem objects can be selected. (Inherited from *FlexChart*)
showDataTips	whether DataTip controls for the chart show. (Inherited from *FlexChart*)
showDataTipTargets	whether to show targets over the datapoints when showDataTips is set to true. (Inherited from *FlexChart*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexCartesianChart*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAxisRatio	the width limit of the vertical axis. (Inherited from *FlexCartesianChart*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexChart*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexChart*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexChart*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChart*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Defines the value of the type property of the event object for a headerRelease event, which indicates that the user pressed and released the mouse on a column header. (Inherited from *FlexChart*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexPlotSeries Class

Description
Defines a data series for a PlotChart control.

Inheritance Hierarchy

- *FlexChartSeries*
  - FlexPlotSeries

Syntax

```
'Declaration
Public Class FlexPlotSeries _
Inherits FlexChartSeries
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexChartSeries*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexChartSeries*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)

Name	Description
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexChartSeries*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexChartSeries*)
numRows	the total number of rows of data available. (Inherited from *FlexChartSeries*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
radius	the radius style
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the selected item in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedIndices	an array of indexes of the selected items in the data provider of the series. (Inherited from *FlexChartSeries*)
selectedItem	the chart item that is selected in the series. (Inherited from *FlexChartSeries*)
selectedItems	an array of chart items that are selected in the series. (Inherited from *FlexChartSeries*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is Nothing. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
xField	the field of the data provider that determines the x-axis location of each data point.
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
yField	the field of the data provider that determines the y-axis location of each data point.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexChartSeries*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SeriesClick	Indicates that the user clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesDoubleClick	Indicates that the user double-clicked the mouse button over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SeriesItemRollOver	Indicates that the user rolled the mouse pointer over a chart item representing data in the chart. (Inherited from *FlexChartSeries*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by

Name	Description
	the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexPopUpButton Class

Description

The PopUpButton control adds a flexible pop-up control interface to a Button control. It contains a main button and a secondary button, called the pop-up button, which pops up any UIComponent object when a user clicks the pop-up button.

A PopUpButton control can have a text label, an icon, or both on its face. When a user clicks the main part of the PopUpButton control, it dispatches a click event.

One common use for the PopUpButton control is to have the pop-up button open a List control or a Menu control that changes the function and label of the main button.

Inheritance Hierarchy

- *FlexButton*
  - FlexPopUpButton

Syntax

```
'Declaration
Public Class FlexPopUpButton _
Inherits FlexButton
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. The default value is 0x0B333C. (Inherited from *FlexButton*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3. (Inherited from *FlexButton*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alphas used for the background fill of controls. (Inherited from *FlexButton*)
fillColors	the colors used to tint the background of the control. (Inherited from *FlexButton*)
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4. (Inherited from *FlexButton*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexButton*)
fontSize	the height of the text, in pixels. (Inherited from *FlexButton*)
fontStyle	whether the text is italic font. (Inherited from *FlexButton*)
fontWeight	whether the text is boldface. (Inherited from *FlexButton*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the Button control. (Inherited from *FlexButton*)

Name	Description
labelPlacement	the orientation of the label in relation to a specified icon. (Inherited from *FlexButton*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	whether a toggle button is toggled on (true) or off (false). (Inherited from *FlexButton*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexButton*)
textDecoration	whether the text is underlined. (Inherited from *FlexButton*)
textIndent	the offset of the first line of text from the left side of the container, in pixels. The default value is 0. (Inherited from *FlexButton*)
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C. (Inherited from *FlexButton*)
textSelectedColor	the text color of the label as the user presses it. The default value is 0x000000. (Inherited from *FlexButton*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)

Name	Description
toggle	whether a Button is in a toggle state or not. (Inherited from *FlexButton*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Dispatched when the specified UIComponent closes.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
Open	Dispatched when the specified UIComponent opens.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexButton*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexProgressBar Class

Description

The ProgressBar control provides a visual representation of the progress of a task over time. There are two types of ProgressBar controls: determinate and indeterminate.

A determinate ProgressBar control is a linear representation of the progress of a task over time. You use a determinate ProgressBar when the scope of the task is known. It displays when the user has to wait for an extended amount of time.

An indeterminate ProgressBar control represents time-based processes for which the scope is not yet known. As soon as you can determine the scope, you should use a determinate ProgressBar control.

Inheritance Hierarchy

- *FlexObject*
  - FlexProgressBar

Syntax

```
'Declaration
Public Class FlexProgressBar _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)

Name	Description
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
barColor	the color of a ProgressBar.
borderColor	the black section of a three-dimensional border, or the color section of a two-dimensional border. The default value is 0xB7BABC.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. The default value is 0x0B333C.
conversion	the number used to convert incoming current bytes loaded value and the total bytes loaded values.
currentState	the current view state of the component. (Inherited from *FlexObject*)
direction	the direction in which the fill of the ProgressBar expands toward completion. Valid values in MXML are "right" and "left".
disabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillColors	the colors used to tint the background fill of the component.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels.
fontStyle	whether the text is italic font.
fontWeight	whether the text is boldface.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
indeterminate	whether the ProgressBar control has a determinate or indeterminate appearance.
label	the text that accompanies the progress bar.
labelPlacement	the placement of the label.

Name	Description
labelWidth	the width of the label in pixels.
maximum	the largest progress value for the ProgressBar. You can only use this property in manual mode.
minimum	the smallest progress value for the ProgressBar. This property is set by the developer only in manual mode.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentComplete	the percentage of process that is completed. The range is 0 to 100.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container.
textDecoration	whether the text is underlined.
textIndent	the offset of first line of text from the left side of the container, in pixels. The default value is 0.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	the amount of progress that has been made between the minimum and maximum values.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexRadioButton Class

Description
The RadioButton control lets the user make a single choice within a set of mutually exclusive choices.

Inheritance Hierarchy

- *FlexButton*
  - FlexRadioButton

Syntax
```
'Declaration
Public Class FlexRadioButton _
Inherits FlexButton
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. The default value is 0x0B333C. (Inherited from *FlexButton*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3. (Inherited from *FlexButton*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)

Name	Description
fillAlphas	the alphas used for the background fill of controls. (Inherited from *FlexButton*)
fillColors	the colors used to tint the background of the control. (Inherited from *FlexButton*)
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4. (Inherited from *FlexButton*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexButton*)
fontSize	the height of the text, in pixels. (Inherited from *FlexButton*)
fontStyle	whether the text is italic font. (Inherited from *FlexButton*)
fontWeight	whether the text is boldface. (Inherited from *FlexButton*)
groupName	the name of the group to which this RadioButton control belongs, or specifies the value of the id property of a RadioButtonGroup control if this RadioButton is part of a group defined by a RadioButtonGroup control.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the Button control. (Inherited from *FlexButton*)
labelPlacement	the orientation of the label in relation to a specified icon. (Inherited from *FlexButton*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)

Name	Description
selected	whether a toggle button is toggled on (true) or off (false). (Inherited from *FlexButton*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textAlign	the alignment of text within a container. (Inherited from *FlexButton*)
textDecoration	whether the text is underlined. (Inherited from *FlexButton*)
textIndent	the offset of the first line of text from the left side of the container, in pixels. The default value is 0. (Inherited from *FlexButton*)
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C. (Inherited from *FlexButton*)
textSelectedColor	the text color of the label as the user presses it. The default value is 0x000000. (Inherited from *FlexButton*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toggle	whether a Button is in a toggle state or not. (Inherited from *FlexButton*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)

Name	Description
Type	Dispatched when the user presses a key. (Inherited from *FlexButton*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexRepeater Class

Description
Identifies the repeater control, which creates multiple instances of its subcomponents.

Inheritance Hierarchy

- *TestObject*
  - FlexRepeater

Syntax

```
'Declaration
Public Class FlexRepeater _
```

```
Inherits TestObject _
Implements IClickable, IKeyable
```

Properties

Name	Description
automationClassName	a string containing the automation class name for the element.
automationIndex	a string containing the automation index for the element.
automationName	the name that can be used as an identifier for this object.
className	a string containing the class name of the element as assigned by the control developer.
columnNames	a list containing the names of all columns in the data.
count	the number of times this Repeater should execute.
firstVisibleRow	the index of the first visible child.
id	a string containing the automation identifier (ID) for the element.
lastVisibleRow	the index of the last visible child.
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
recycleChildren	whether this Repeater should re-use previously created children, or create new ones.
startingIndex	the index into the dataProvider at which this Repeater starts creating children.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexRule Class

Description

Creates a single horizontal or vertical line. You typically use this control to create a dividing line within a container.

Inheritance Hierarchy

- *FlexObject*
  - FlexRule

Syntax

```
'Declaration
Public Class FlexRule _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of the line.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)

Name	Description
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
strokeWidth	the thickness of the rule in pixels.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexScrollBar Class

Description
Lets you control the portion of data that is displayed when there is too much data to fit in a display area.

Inheritance Hierarchy

- *FlexObject*
  - FlexScrollBar

Syntax
```
'Declaration
Public Class FlexScrollBar _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillColors	the colors used to tint the background of the control.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)

Name	Description
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lineScrollSize	the amount to scroll when an arrow button is pressed, in pixels. The default value is 1.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
pageScrollSize	the amount to move the scroll thumb when the scroll bar track is pressed, in pixels. The default value is 0.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
scrollPosition	the number that represents the current scroll position.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
trackColors	The colors of the track, as an array of two colors.
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the ScrollBar control scrolls through user initiated action or programmatically.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexScrollBase Class

Description

The ScrollControlBase class is the base class for controls with scroll bars.

The user interacts with the scroll bar or the developer accesses methods and properties that alter the viewable area. The ScrollControlBase takes a single child object and positions and masks or sizes that object to display the viewable content. All items to be scrolled must be children of that content object.

Inheritance Hierarchy

- *FlexObject*
  - FlexScrollBase
    - *FlexListBase*
    - *FlexTextArea*
    - *SparkAirHTML*

Syntax

```
'Declaration
Public Class FlexScrollBase _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)

Name	Description
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance.
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexSlider Class

Description

Lets users select a value by moving a slider thumb between the end points of the slider track.

The current value of the slider is determined by the relative location of the thumb between the end points of the slider, corresponding to the slider's minimum and maximum values.

Inheritance Hierarchy

- *FlexObject*
  - FlexSlider

Syntax

```
'Declaration
Public Class FlexSlider _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipPrecision	the number of decimal places to use for the data tip text.
direction	the orientation of the slider control.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)

Name	Description
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillColors	the colors used to tint the background of the control.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
labels	an array of strings used for the slider labels.
labelStyleName	the name of the style to use for the slider label. The default value is undefined.
liveDragging	whether live dragging is enabled for the slider.
maximum	the maximum allowed value on the slider. The default value is 10.
minimum	the minimum allowed value on the slider control. The default value is 0.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
showDataTip	whether the data tip shows during user interaction.
snapInterval	the increment value of the slider thumb as the user moves the thumb.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)

Name	Description
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
thumbCount	the number of thumbs allowed on the slider.
tickColor	the color of the tick marks.
tickLength	the length in pixels of the tick marks.
tickThickness	the thickness in pixels of the tick marks.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
trackColors	the colors of the track, as an array of two colors.
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	the position of the thumb, and is a number between the minimum and maximum properties.
values	an array of values for each thumb when thumbCount is greater than 1.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Change	Dispatched when the slider changes value due to mouse or keyboard interaction.
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexStandalonePlayer Class

Description
The Flex Standalone Player.

Inheritance Hierarchy

- *Window*
  - `FlexStandalonePlayer`

Syntax

```
'Declaration
Public Class FlexStandalonePlayer _
Inherits Window
```

Properties

Name	Description
Application	The name of the Application that this Window belongs to. (Inherited from *Window*)

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexTabNavigator Class

Description
Extends the ViewStack container by including a TabBar container for navigating between its child containers.

Inheritance Hierarchy

- *FlexViewStack*
  - FlexTabNavigator

Syntax

```
'Declaration
Public Class FlexTabNavigator _
Inherits FlexViewStack
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAlign	the horizontal positioning of tabs at the top of this TabNavigator container.
horizontalGap	the separation between tabs, in pixels. The default value is -1, so that the borders of adjacent tabs overlap.
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)

Name	Description
selectedChild	a reference to the currently visible child container. (Inherited from *FlexViewStack*)
selectedIndex	the zero-based index of the currently visible child container. (Inherited from *FlexViewStack*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabHeight	the height of each tab, in pixels.
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
tabWidth	the width of each tab, in pixels.
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Change	Defines the value of the type property of the event object for an itemClick event.
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)

Name	Description
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexViewStack*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexTextArea Class

Description

A multiline text field with a border and optional scroll bars.

The TextArea control supports the HTML rendering capabilities of Flash Player and AIR.

Inheritance Hierarchy

- *FlexScrollBase*
  - FlexTextArea

Syntax

```
'Declaration
Public Class FlexTextArea _
Inherits FlexScrollBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
color	the color of text in the component, including the component label. The default value is 0x0B333C.
currentState	the current view state of the component. (Inherited from *FlexObject*)
disabledColor	the background color of a component.
displayAsPassword	whether this control is used for entering passwords.
editable	whether the user can edit the text in this control.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use.
fontSize	the height of the text, in pixels.
fontStyle	whether the text is italic font.
fontWeight	whether the text is boldface.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalScrollPolicy	whether the horizontal scroll bar is always on (ScrollPolicy.ON), always off (ScrollPolicy.OFF), or turns on when needed (ScrollPolicy.AUTO).
htmlText	the text displayed by the TextInput control, including HTML markup that expresses the styles of that text.
id	the ID of the component. (Inherited from *FlexObject*)
length	the number of characters of text displayed in the TextArea.
maxChars	the maximum number of characters that users can enter in the text field.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
restrict	the set of characters that a user can enter into the control.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectionBeginIndex	the zero-based character index value of the first character in the current selection.
selectionEndIndex	the zero-based index of the position after the last character in the current selection (equivalent to the one-based index of the last character).

Name	Description
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	the plain text that appears in the control.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
wordWrap	whether text in the row should be word wrapped.
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Dispatched when the user types, deletes, or pastes text into the control.

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexScrollBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event.
SelectText	Lets you track selection within a text field.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexTitleWindow Class

Description

Contains a title bar, a caption, a border, and a content area for its child.

Typically, you use TitleWindow containers to wrap self-contained application modules. For example, you can include a form in a TitleWindow container. When the user completes the form, you can close the TitleWindow container programmatically, or let the user close it by using the Close button.

Inheritance Hierarchy

- *FlexPanel*
  - FlexTitleWindow

Syntax

```
'Declaration
Public Class FlexTitleWindow _
Inherits FlexPanel
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dropShadowEnabled	whether the Panel container's drop shadow is visible. The default value is true. (Inherited from *FlexPanel*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)

Name	Description
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
footerColors	the array of two colors used to draw the footer (area for the ControlBar container) background. (Inherited from *FlexPanel*)
headerColors	the array of two colors used to draw the header. (Inherited from *FlexPanel*)
headerHeight	the height of the header. The default value is based on the style of the title text. (Inherited from *FlexPanel*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is

Name	Description
	NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
showCloseButton	whether to display a Close button in the TitleWindow container. The default value is false.\
status	the text in the status area of the title bar. The default value is "". (Inherited from *FlexPanel*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
title	the title or caption displayed in the title bar. The default value is "". (Inherited from *FlexPanel*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexToggleButtonBar Class

Description

Defines a horizontal or vertical group of buttons that maintain their selected or deselected state.

Only one button in the ToggleButtonBar control can be in the selected state. This means that when a user selects a button in a ToggleButtonBar control, the button stays in the selected state until the user selects a different button.

Inheritance Hierarchy

- *FlexButtonBar*
  - FlexToggleButtonBar

Syntax

```
'Declaration
Public Class FlexToggleButtonBar _
Inherits FlexButtonBar
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)

Name	Description
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
direction	whether to use vertical (default) or horizontal layout. (Inherited from *FlexBox*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAlign	the horizontal alignment of children in the container. (Inherited from *FlexBox*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	the index of the active navigation item, where the first item is at an index of 0. The default value is -1. (Inherited from *FlexNavigationBar*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAlign	the vertical alignment of children in the container. (Inherited from *FlexBox*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)

Name	Description
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Defines the value of the type property of the event object for an itemClick event. (Inherited from *FlexNavigationBar*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexNavigationBar*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use

Name	Description
	WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexTree Class

Description

The Tree control lets a user view hierarchical data arranged as an expandable tree.

Each item in a tree can be a leaf or a branch. A leaf item is an end point in the tree. A branch item can contain leaf or branch items, or it can be empty.

By default, a leaf is represented by a text label next to a file icon. A branch is represented by a text label next to a folder icon, with a disclosure triangle that a user can open to expose children.

Inheritance Hierarchy

- *FlexList*
  - FlexTree

Syntax

```
'Declaration
Public Class FlexTree _
Inherits FlexList
```

Properties

Name	Description
allowMultipleSelection	whether you can allow more than one item to be selected at the same time. (Inherited from *FlexListBase*)
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alternatingItemColors	the colors to use for the backgrounds of the items in the list. (Inherited from *FlexListBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)

Name	Description
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundDisabledColor	the color of text in the component if it is disabled. The default value is 0xAAB3B3. (Inherited from *FlexListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnCount	the number of columns to be displayed in a TileList control or items in a HorizontalList control. (Inherited from *FlexListBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexListBase*)
columnWidth	the width of the control's columns. (Inherited from *FlexListBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dataTipField	the name of the field in the data provider items to display as the data tip. (Inherited from *FlexListBase*)
depthColors	the array of colors used in the Tree control, in descending order. The default value is undefined.
disabledColor	the disabled color of a list item. The default value is 0xDDDDDD.
editable	whether the user can edit items in the data provider.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexListBase*)
focusAlpha	the alpha transparency value of the focus skin. The default value is 0.4. (Inherited from *FlexListBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
iconField	the name of the field in the data provider object that determines what to display as the icon. (Inherited from *FlexListBase*)
id	the ID of the component. (Inherited from *FlexObject*)
indentation	the indentation for each tree level, in pixels. The default value is 17.
labelField	the name of the field in the data provider items to display as the label. (Inherited from *FlexListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexListBase*)

Name	Description
lockedColumnCount	the index of the first column in the control that scrolls. (Inherited from *FlexListBase*)
lockedRowCount	the index of the first row in the control that scrolls. (Inherited from *FlexListBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *FlexListBase*)
numRows	the total number of rows of data available. (Inherited from *FlexListBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
rollOverColor	the color of the background of a renderer when the user rolls over it. The default value is 0xEEFEE6. (Inherited from *FlexListBase*)
rowCount	the number of rows to be displayed. (Inherited from *FlexListBase*)
rowHeight	the height of the rows in pixels. (Inherited from *FlexListBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	whether the list shows selected items as selected. (Inherited from *FlexListBase*)
selectedIndex	the index in the data provider of the selected item. (Inherited from *FlexListBase*)
selectedIndices	an array of indices in the data provider of the selected items. (Inherited from *FlexListBase*)
selectedItem	a reference to the selected item in the data provider. (Inherited from *FlexListBase*)
selectedItems	an array of references to the selected items in the data provider. (Inherited from *FlexListBase*)

Name	Description
selectionColor	the color of the background of a renderer when the user selects it. The default value is 0x7FCEFF. (Inherited from *FlexListBase*)
selectionDisabledColor	the color of the background of a renderer when the component is disabled. The default value is 0xDDDDDD. (Inherited from *FlexListBase*)
showDataTips	whether dataTips are displayed for text in the rows. (Inherited from *FlexListBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
textRollOverColor	the text color of the label as the user moves the mouse pointer over the button. The default value is 0x2B333C. (Inherited from *FlexListBase*)
textSelectedColor	the color of the text of a renderer when the user selects it. The default value is 0x2B333C. (Inherited from *FlexListBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useRollOver	whether items are highlighted as the mouse rolls over them. (Inherited from *FlexListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
variableRowHeight	whether the individual rows can have different height. (Inherited from *FlexListBase*)
verticalAlign	the vertical alignment of a renderer in a row. (Inherited from *FlexListBase*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
wordWrap	whether text in the row should be word wrapped. (Inherited from *FlexListBase*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Name	Description
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes or collapses a tree branch.
Deselect	Defines the value of the type property of the event object for an event that is dispatched when a previously selected item is deselected. (Inherited from *FlexListBase*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Defines the value of the type property of the ListEvent object for an itemDoubleClick event, which indicates that the user double-clicked the mouse over a visual item in the control. (Inherited from *FlexListBase*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexListBase*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexListBase*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexListBase*)
Edit	Defines the value of the type property of the event object for a itemEditBegin event, which indicates that an item is ready to be edited. (Inherited from *FlexList*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *FlexListBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexScrollBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
MultiSelect	Defines the value of the type property of the event object for an event that is dispatched when an item is selected as part of an action that selects multiple items. (Inherited from *FlexListBase*)
Open	Opens or expands a tree branch.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Defines the value of the type property of the event object for a scroll event. (Inherited from *FlexListBase*)
ScrollToIndex	Ensures that the data provider item at the given index is visible. (Inherited from *FlexListBase*)
Select	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)
SelectIndex	Defines the value of the type property of the event object for an event that is dispatched when a single item is selected. (Inherited from *FlexListBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *FlexListBase*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexUIMovieClip Class

Description

Components created in Adobe Flash CS3 Professional for use in Flex are subclasses of the mx.flash.UIMovieClip class.

The UIMovieClip class implements the interfaces necessary for a Flash component to be used like a normal Flex component. Therefore, a subclass of UIMovieClip can be used as a child of a Flex container or as a skin, and it can respond to events, define view states and transitions, and work with effects in the same way as can any Flex component.

Inheritance Hierarchy

- *FlexDisplayObject*
  - FlexUIMovieClip

- *FlexContainerMovieClip*

Syntax

```
'Declaration
Public Class FlexUIMovieClip _
Inherits FlexDisplayObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element.
automationIndex	a string containing the automation index for the element.
automationName	the name that can be used as an identifier for this object.
className	the name of this instance's class, such as "Button".
currentState	the current state of this component.
enabled	whether a movie clip is enabled.
errorColor	the color of the component highlight when validation fails.
errorString	the text that is displayed by a component's error tip when a component is monitored and validation fails.
focusEnabled	whether the component can receive focus when selected.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. Valid values are 0 to 100.
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Valid values are 0 to 100.
scaleX	the horizontal scale (percentage) of the object as applied from the registration point. The default registration point is (0,0). 1.0 equals 100% scale.
scaleY	the vertical scale (percentage) of an object as applied from the registration point of the object. The default registration point is (0,0). 1.0 is 100% scale.

Name	Description
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component.
toolTip	the text to display in the ToolTip. The default value is `Nothing`.
useHandCursor	whether the pointing hand (hand cursor) appears when the mouse rolls over a sprite in which the buttonMode property is set to true.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Defines the value of the type property of a keyFocusChange event object.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Called by the FocusManager when the component receives focus. The component may in turn set focus to an internal component.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexVideoDisplay Class

Description

Lets you play an FLV file in a Flex application. It supports progressive download over HTTP, streaming from the Flash Media Server, and streaming from a Camera object.

Inheritance Hierarchy

- *FlexObject*
  - FlexVideoDisplay

Syntax

```
'Declaration
Public Class FlexVideoDisplay _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)

Name	Description
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
source	the relative path and filename of the FLV file to stream.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
totalTime	the total length of the media, in seconds.
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from FlexObject)
Value	The value of the control, e.g.: text in a text control. (Inherited from TestObject)
visible	whether the display object is visible. (Inherited from FlexDisplayObject)
volume	the volume level, specified as an value between 0 and 1. The default value is 0.75.
width	the width of the display object, in pixels. (Inherited from FlexDisplayObject)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from FlexDisplayObject)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from FlexDisplayObject)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from TestObject)
ChangeFocus	Changes the current focus. (Inherited from FlexObject)
Click	Clicks on the object. (Inherited from IClickable)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from IClickable)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from TestObject)
Exists	Checks if an object exists in the application under test. (Inherited from TestObject)
Find	Finds an object specified by an XPath locator. (Inherited from TestObject)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a

Name	Description
	WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexViewStack Class

Description
Consists of a collection of child containers stacked on top of each other, where only one child at a time is visible. When a different child container is selected, it seems to replace the old one because it appears in the same location. However, the old child container still exists; it is just invisible.

Inheritance Hierarchy

- *FlexContainer*
  - FlexViewStack
    - *FlexTabNavigator*

Syntax

```
'Declaration
Public Class FlexViewStack _
Inherits FlexContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)

Name	Description
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)

Name	Description
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedChild	a reference to the currently visible child container.
selectedIndex	the zero-based index of the currently visible child container.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown.

Name	Description
	The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexWindow Class

Description
The Window is a top-level container for additional windows in an AIR desktop application.

Inheritance Hierarchy

- *FlexContainer*
  - FlexWindow

Syntax

```
'Declaration
Public Class FlexWindow _
Inherits FlexContainer _
Implements IMoveable
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alwaysInFront	a value whether the underlying NativeWindow is always in front of other windows.
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)

Name	Description
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
closed	a value whether the underlying window has been closed.
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)

Name	Description
isAIRWindow	a value whether this window is an AIR window.
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
maxHeight	The maximum recommended height of the component to be considered by the parent during layout.
maximizable	a value whether the window can be maximized.
maxWidth	The maximum recommended width of the component to be considered by the parent during layout.
minHeight	The minimum recommended height of the component to be considered by the parent during layout.
minimizable	a value whether the window can be minimized.
minWidth	The minimum recommended width of the component to be considered by the parent during layout.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizable	a value whether the window can be resized.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
showStatusBar	a value whether the status bar is visible.

Name	Description
status	The string that appears in the status bar, if it is visible.
systemChrome	the type of system chrome (if any) the window has.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
title	The title that appears in the window title bar and the taskbar.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
transparent	a value whether the window is transparent.
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *IMoveable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
StateChange	Dispatched when the displaystatus of the window is changed.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use

Name	Description
	WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FlexWindowedApplication Class

Description
The WindowedApplication defines the application container that you use to create Flex applications for AIR applications.

Inheritance Hierarchy

- *FlexApplication*
  - FlexWindowedApplication

Syntax

```
'Declaration
Public Class FlexWindowedApplication _
Inherits FlexApplication _
Implements IMoveable
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alwaysInFront	a value whether the underlying NativeWindow is always in front of other windows.
applicationID	The identifier that AIR uses to identify the application.
autoExit	a value whether the AIR application will quit when the last window closes or will continue running in the background.
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)

Name	Description
backgroundAlpha	the alpha level of the color defined by the backgroundColor property, of the image or SWF file defined by the backgroundImage style. Valid values range from 0.0 to 1.0. (Inherited from *FlexContainer*)
backgroundColor	the background color of a component. (Inherited from *FlexContainer*)
backgroundDisabledColor	the background color of the component when it is disabled. The global default value is undefined. (Inherited from *FlexContainer*)
backgroundImage	the background image of a component. This can be an absolute or relative URL or class. (Inherited from *FlexContainer*)
backgroundSize	the percentage to change the image size to for the specified backgroundImage. (Inherited from *FlexContainer*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
closed	a value whether the underlying window has been closed.
columnNames	a list containing the names of all columns in the data. (Inherited from *FlexContainer*)
creationIndex	the order to instantiate and draw the children of the container. (Inherited from *FlexContainer*)
creationPolicy	the child creation policy for this Container. (Inherited from *FlexContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillAlphas	the alpha transparency values used for the background fill of components. (Inherited from *FlexContainer*)
fillColors	the colors used to tint the background fill of the component. (Inherited from *FlexContainer*)
firstVisibleRow	the index of the first visible child. (Inherited from *FlexContainer*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
fontFamily	the name of the font to use. (Inherited from *FlexContainer*)
fontSize	the height of the text, in pixels. (Inherited from *FlexContainer*)
fontStyle	whether the text is italic font. (Inherited from *FlexContainer*)
fontWeight	whether the text is boldface. (Inherited from *FlexContainer*)

Name	Description
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text displayed by some navigator containers to represent this Container. (Inherited from *FlexContainer*)
lastVisibleRow	the index of the last visible child. (Inherited from *FlexContainer*)
maxHeight	The maximum recommended height of the component to be considered by the parent during layout.
maximizable	a value whether the window can be maximized.
maxWidth	The maximum recommended width of the component to be considered by the parent during layout.
minHeight	The minimum recommended height of the component to be considered by the parent during layout.
minimizable	a value whether the window can be minimized.
minWidth	The minimum recommended width of the component to be considered by the parent during layout.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numChildren	the number of child components in this container. (Inherited from *FlexContainer*)
numColumns	the total number of columns of data available. (Inherited from *FlexContainer*)
numRows	the total number of rows of data available. (Inherited from *FlexContainer*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizable	a value whether the window can be resized.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)

Name	Description
showStatusBar	a value whether the status bar is visible.
status	The string that appears in the status bar, if it is visible.
systemChrome	the type of system chrome (if any) the window has.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
title	The title that appears in the window title bar and the taskbar.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
transparent	a value whether the window is transparent.
url	the URL from which this Application's SWF file was loaded. (Inherited from *FlexApplication*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Defines the value of the type property of the event object for a dragComplete event. (Inherited from *FlexContainer*)
DragDrop	Defines the value of the type property of the event object for a dragDrop event. (Inherited from *FlexContainer*)
DragStart	Defines the value of the type property of the event object for a dragStart event. (Inherited from *FlexContainer*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. The return value is an array of rows, each of which is an array of items. (Inherited from *FlexContainer*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexContainer*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *IMoveable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
Scroll	Dispatched when the user manually scrolls the container. (Inherited from *FlexContainer*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkAirHTML Class

Description
The HTML control lets you display HTML content in your application.

Inheritance Hierarchy
- *FlexScrollBase*
  - SparkAirHTML

Syntax
```
'Declaration
Public Class SparkAirHTML _
Inherits FlexScrollBase
```

Properties
Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)

Name	Description
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when a mouse wheel is spun over an InteractiveObject instance. (Inherited from *FlexScrollBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the scroll events occur. This event is dispatched if the scrolling is done programmatically or by the user.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a

Name	Description
	WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkApplication Class

Description
Flex defines a default, or Application, container that lets you start adding content to your application without explicitly defining another container.

Inheritance Hierarchy

- *SparkSkinnableContainer*
  - SparkApplication
    - *SparkWindowedApplication*

Syntax

```
'Declaration
Public Class SparkApplication _
Inherits SparkSkinnableContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
colorCorrection	The value of the stage's colorCorrection property.

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
controlBarVisible	whether the control bar is visible.
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)

Name	Description
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
url	The URL from which this Application's SWF file was loaded.
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
viewSourceURL	the URL where the application's source can be viewed.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha

Name	Description
	channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)

Name	Description
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkBorderContainer Class

Description

The BorderContainer class defines a set of CSS styles that control the appearance of the border and background fill of the container.

Inheritance Hierarchy

- *SparkSkinnableContainer*

- SparkBorderContainer

Syntax

```
'Declaration
Public Class SparkBorderContainer _
Inherits SparkSkinnableContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)

Name	Description
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkButton Class

Description

The SparkButton component is a commonly used rectangular button. The SparkButton component looks like it can be pressed.

Inheritance Hierarchy

- *SparkButtonBase*
  - SparkButton
    - *SparkMuteButton*

Syntax

```
'Declaration
Public Class SparkButton _
Inherits SparkButtonBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
autoRepeat	whether to dispatch repeated buttonDown events if the user holds down the mouse button. (Inherited from *SparkButtonBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
emphasized	whether the default button as requested by the focus manager.

Name	Description
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the ButtonBase control. (Inherited from *SparkButtonBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
stickyHighlighting	a value whether the button displays its down skin when the user presses it but changes to its over skin when the user drags the mouse off of it. (Inherited from *SparkButtonBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkButtonBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use

Name	Description
	WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkButtonBar Class

Description
The ButtonBar control defines a horizontal group of logically related buttons with a common look and navigation.

Inheritance Hierarchy

- *SparkListBase*
  - SparkButtonBar

Syntax

```
'Declaration
Public Class SparkButtonBar _
Inherits SparkListBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	the measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableDataContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
caretIndex	The item that is currently in focus. (Inherited from *SparkListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
labelField	The name of the field in the data provider items to display as the label. (Inherited from *SparkListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
requireSelection	a value if a data item must always be selected in the control (Inherited from *SparkListBase*)

Name	Description
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	The 0-based index of the selected item, or -1 if no item is selected. (Inherited from *SparkListBase*)
selectedItem	The item that is currently selected. (Inherited from *SparkListBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useVirtualLayout	the value of the useVirtualLayout property of the layout associated with this control. (Inherited from *SparkListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha

Name	Description
	channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemsCount	Gets the number of items in the ComboBox. (Inherited from *SparkListBase*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Dispatched when an item in the list is selected. (Inherited from *SparkListBase*)
SelectIndex	Dispatched when the user clicks on an item in the list or navigates to the item using a keyboard.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkButtonBarButton Class

Description
The ButtonBarButton class defines the custom item renderer used by the ButtonBar control.

Inheritance Hierarchy

- *SparkToggleButton*
  - SparkButtonBarButton

Syntax
```
'Declaration
Public Class SparkButtonBarButton _
Inherits SparkToggleButton
```

Properties

Name	Description
allowDeselection	a value whether the user click on a currently selected button deselects it.
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
autoRepeat	whether to dispatch repeated buttonDown events if the user holds down the mouse button. (Inherited from *SparkButtonBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the ButtonBase control. (Inherited from *SparkButtonBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	whether the button is in the down state, and false if it is in the up state. (Inherited from *SparkToggleButtonBase*)
showsCaret	a value whether the item renderer can show itself as focused.
stickyHighlighting	a value whether the button displays its down skin when the user presses it but changes to its over skin when the user drags the mouse off of it. (Inherited from *SparkButtonBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled. (Inherited from *SparkToggleButton*)
DragDrop	The event triggered when the dragged item is dropped. (Inherited from *SparkToggleButton*)
DragStart	The event triggered when the drag/drop operation begins. (Inherited from *SparkToggleButton*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkButtonBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkButtonBase Class

Description
The SparkButtonBase class is the base class for the all Spark button components.

Inheritance Hierarchy

- *SparkObject*
  - SparkButtonBase
    - *SparkButton*
    - *SparkToggleButtonBase*

Syntax
```
'Declaration
Public Class SparkButtonBase _
Inherits SparkObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)

Name	Description
autoRepeat	whether to dispatch repeated buttonDown events if the user holds down the mouse button.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the ButtonBase control.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
stickyHighlighting	a value whether the button displays its down skin when the user presses it but changes to its over skin when the user drags the mouse off of it.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)

Name	Description
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkCheckBox Class

Description

The CheckBox component consists of an optional label and a small box that can contain a check mark or not.

Inheritance Hierarchy

- *SparkToggleButtonBase*
  - SparkCheckBox

Syntax

```
'Declaration
Public Class SparkCheckBox _
Inherits SparkToggleButtonBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
autoRepeat	whether to dispatch repeated buttonDown events if the user holds down the mouse button. (Inherited from *SparkButtonBase*)

Name	Description
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the ButtonBase control. (Inherited from *SparkButtonBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	whether the button is in the down state, and false if it is in the up state. (Inherited from *SparkToggleButtonBase*)
stickyHighlighting	a value whether the button displays its down skin when the user presses it but changes to its over skin when the user drags the mouse off of it. (Inherited from *SparkButtonBase*)

Name	Description
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled.
DragDrop	The event triggered when the dragged item is dropped.

Name	Description
DragStart	The event triggered when the drag/drop operation begins.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkButtonBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkComboBox Class

Description

The ComboBox control is a child class of the DropDownListBase control. Like the DropDownListBase control, when the user selects an item from the drop-down list in the ComboBox control, the data item appears in the prompt area of the control.

Inheritance Hierarchy

- *SparkDropDownListBase*
  - SparkComboBox

Syntax

```
'Declaration
Public Class SparkComboBox _
Inherits SparkDropDownListBase
```

Properties

Name	Description
allowMultipleSelection	a value if multiple selection is enabled. (Inherited from *SparkList*)
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)

Name	Description
autoLayout	the measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableDataContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
caretIndex	The item that is currently in focus. (Inherited from *SparkListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dragEnabled	A flag that indicates whether you can drag items out of this control and drop them on other controls. (Inherited from *SparkList*)
dragMoveEnabled	A flag that indicates whether items can be moved instead of just copied from the control as part of a drag-and-drop operation. (Inherited from *SparkList*)
dropEnabled	A flag that indicates whether dragged items can be dropped onto the control. (Inherited from *SparkList*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
isDropDownOpen	a value whether the drop down is open. (Inherited from *SparkDropDownListBase*)
labelField	The name of the field in the data provider items to display as the label. (Inherited from *SparkListBase*)

Name	Description
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
maxChars	The maximum number of characters that the prompt area can contain, as entered by a user.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
requireSelection	a value if a data item must always be selected in the control (Inherited from *SparkListBase*)
restrict	The set of characters that a user can enter into the prompt area.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	The 0-based index of the selected item, or -1 if no item is selected. (Inherited from *SparkListBase*)
selectedIndices	a list representing the indices of the currently selected item or items. (Inherited from *SparkList*)
selectedItem	The item that is currently selected. (Inherited from *SparkListBase*)
selectedItems	a list of Objects representing the currently selected data items. (Inherited from *SparkList*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)

Name	Description
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useVirtualLayout	the value of the useVirtualLayout property of the layout associated with this control. (Inherited from *SparkListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)

Name	Description
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemsCount	Gets the number of items in the ComboBox. (Inherited from *SparkListBase*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Dispatched when the user types, deletes, or pastes text into the control.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
Open	Dispatched when the user clicks the drop-down button to display the drop-down list. (Inherited from *SparkDropDownListBase*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Dispatched when an item in the list is selected. (Inherited from *SparkListBase*)
SelectIndex	Dispatched when the user clicks on an item in the list or navigates to the item using a keyboard. (Inherited from *SparkList*)
SelectText	Dispatched when text is selected.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkComplexDisplay Class

Description
A complex rendered object.

Inheritance Hierarchy

- *SparkListLabel*

- SparkComplexDisplay

Syntax

```
'Declaration
Public Class SparkComplexDisplay _
Inherits SparkListLabel
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	a value whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkGroupBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
automationValue	A value which can be used to uniquely identify this item. (Inherited from *SparkListLabel*)
blendMode	A value from the BlendMode class that specifies which blend mode to use. (Inherited from *SparkGroup*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipAndEnableScrolling	a value whether to clip the children to the boundaries of the viewport. (Inherited from *SparkGroupBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkGroupBase*)
contentHeight	The height of the viewport's content. (Inherited from *SparkGroupBase*)
contentWidth	The width of the viewport's contents. (Inherited from *SparkGroupBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dragging	a value whether the SparkListLabel is in dragging mode. (Inherited from *SparkListLabel*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)

Name	Description
firstVisibleRow	the index of the first visible child. (Inherited from *SparkGroupBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalScrollPosition	The x coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component (Inherited from *SparkGroupBase*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the label of the SparkListLabel. (Inherited from *SparkListLabel*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkGroupBase*)
luminosityClip	a value whether the luminosity mask clips the masked content. (Inherited from *SparkGroupBase*)
luminosityInvert	a value that controls the calculation of the RGB color value of a graphic element being masked by a luminosity mask. (Inherited from *SparkGroupBase*)
maskType	The mask type. (Inherited from *SparkGroupBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseEnabledWhereTransparent	a value whether the entire bounds of the Group respond to mouse events such as click and roll over. (Inherited from *SparkGroupBase*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkGroupBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkGroupBase*)
numRows	the total number of rows of data available. (Inherited from *SparkGroupBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)

Name	Description
resizeMode	The ResizeMode for this container. (Inherited from *SparkGroupBase*)
scaleGridBottom	the bottom coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridLeft	the left coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridRight	the right coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridTop	the top coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	a value whether the SparkListLabel is selected. (Inherited from *SparkListLabel*)
showsCaret	a value whether the caret is shown. (Inherited from *SparkListLabel*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalScrollPosition	The y coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component. (Inherited from *SparkGroupBase*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Name	Description
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkGroupBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkGroupBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkGroupBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkGroupBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container. (Inherited from *SparkGroupBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the container's scroll events occur. This event is dispatched if the scrolling is done programmatically or by the user. (Inherited from *SparkGroupBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkGroupBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkDataGrid Class

Description
Identifies a Spark data grid control.

Inheritance Hierarchy

- *SparkSkinnableContainerBase*
  - SparkDataGrid

Syntax

```
'Declaration
Public Class SparkDataGrid _
Inherits SparkSkinnableContainerBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
columnsLength	the count of the columns.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)

Name	Description
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
rowHeight	the height of the row.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedCell	the selected cell.
selectedCells	the selected cells.
selectedIndex	the index of the selected row.
selectedIndices	the indices of the selected rows.

Name	Description
selectedItem	the item of the selected row.
selectedItems	the items of the selected rows.
selectionLength	the number of selected rows.
selectionMode	the selection mode (singlerow, multiplerows, singlecells or multiplecells).
showDataTips	whether data tips should be shown.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
ColumnStretch	Stretches the given column of the data grid.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
EditNext	Sets the following data grid cell in the edit mode.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)

Name	Description
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HeaderClick	Clicks on the specified column header.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects the specified data grid cell.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkDataGridLabel Class

Description
Identifies a spark data grid label control.

Inheritance Hierarchy

- *FlexObject*
  - SparkDataGridLabel

Syntax

```
'Declaration
Public Class SparkDataGridLabel _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
automationValue	A value which can be used to uniquely identify this item.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/

Name	Description
	Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkDataGroup Class

Description
The DataGroup class is the base container class for data items.

Inheritance Hierarchy
- *SparkGroupBase*
  - SparkDataGroup

Syntax
```
'Declaration
Public Class SparkDataGroup _
Inherits SparkGroupBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	a value whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkGroupBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipAndEnableScrolling	a value whether to clip the children to the boundaries of the viewport. (Inherited from *SparkGroupBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkGroupBase*)
contentHeight	The height of the viewport's content. (Inherited from *SparkGroupBase*)
contentWidth	The width of the viewport's contents. (Inherited from *SparkGroupBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkGroupBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalScrollPosition	The x coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component (Inherited from *SparkGroupBase*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkGroupBase*)

Name	Description
luminosityClip	a value whether the luminosity mask clips the masked content. (Inherited from *SparkGroupBase*)
luminosityInvert	a value that controls the calculation of the RGB color value of a graphic element being masked by a luminosity mask. (Inherited from *SparkGroupBase*)
maskType	The mask type. (Inherited from *SparkGroupBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseEnabledWhereTransparent	a value whether the entire bounds of the Group respond to mouse events such as click and roll over. (Inherited from *SparkGroupBase*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkGroupBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkGroupBase*)
numRows	the total number of rows of data available. (Inherited from *SparkGroupBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizeMode	The ResizeMode for this container. (Inherited from *SparkGroupBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalScrollPosition	The y coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component. (Inherited from *SparkGroupBase*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkGroupBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkGroupBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkGroupBase*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkGroupBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container. (Inherited from *SparkGroupBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the container's scroll events occur. This event is dispatched if the scrolling is done programmatically or by the user. (Inherited from *SparkGroupBase*)
Select	Dispatched when one or more items in the list are deselected.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkGroupBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkDataRenderer Class

Description
The SparkDataRenderer class is the base class for data components in spark.

Inheritance Hierarchy
- *SparkGroup*
  - SparkDataRenderer
    - *SparkListLabel*

Syntax
```
'Declaration
Public Class SparkDataRenderer _
Inherits SparkGroup
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	a value whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkGroupBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
blendMode	A value from the BlendMode class that specifies which blend mode to use. (Inherited from *SparkGroup*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipAndEnableScrolling	a value whether to clip the children to the boundaries of the viewport. (Inherited from *SparkGroupBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkGroupBase*)
contentHeight	The height of the viewport's content. (Inherited from *SparkGroupBase*)
contentWidth	The width of the viewport's contents. (Inherited from *SparkGroupBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkGroupBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalScrollPosition	The x coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component (Inherited from *SparkGroupBase*)
id	the ID of the component. (Inherited from *FlexObject*)

Name	Description
lastVisibleRow	the index of the last visible child. (Inherited from *SparkGroupBase*)
luminosityClip	a value whether the luminosity mask clips the masked content. (Inherited from *SparkGroupBase*)
luminosityInvert	a value that controls the calculation of the RGB color value of a graphic element being masked by a luminosity mask. (Inherited from *SparkGroupBase*)
maskType	The mask type. (Inherited from *SparkGroupBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseEnabledWhereTransparent	a value whether the entire bounds of the Group respond to mouse events such as click and roll over. (Inherited from *SparkGroupBase*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkGroupBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkGroupBase*)
numRows	the total number of rows of data available. (Inherited from *SparkGroupBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizeMode	The ResizeMode for this container. (Inherited from *SparkGroupBase*)
scaleGridBottom	the bottom coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridLeft	the left coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridRight	the right coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridTop	the top coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)

Name	Description
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalScrollPosition	The y coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component. (Inherited from *SparkGroupBase*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkGroupBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkGroupBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkGroupBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkGroupBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container. (Inherited from *SparkGroupBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the container's scroll events occur. This event is dispatched if the scrolling is done programmatically or by the user. (Inherited from *SparkGroupBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkGroupBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkDropDownList Class

Description
The SparkDropDownList control contains a drop-down list from which the user can select a single value.

Inheritance Hierarchy

- *SparkDropDownListBase*
  - SparkDropDownList

Syntax

```
'Declaration
Public Class SparkDropDownList _
Inherits SparkDropDownListBase
```

Properties

Name	Description
allowMultipleSelection	a value if multiple selection is enabled. (Inherited from *SparkList*)
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	the measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableDataContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
caretIndex	The item that is currently in focus. (Inherited from *SparkListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dragEnabled	A flag that indicates whether you can drag items out of this control and drop them on other controls. (Inherited from *SparkList*)
dragMoveEnabled	A flag that indicates whether items can be moved instead of just copied from the control as part of a drag-and-drop operation. (Inherited from *SparkList*)
dropEnabled	A flag that indicates whether dragged items can be dropped onto the control. (Inherited from *SparkList*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)

Name	Description
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
isDropDownOpen	a value whether the drop down is open. (Inherited from *SparkDropDownListBase*)
labelField	The name of the field in the data provider items to display as the label. (Inherited from *SparkListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
prompt	The prompt for the DropDownList control.
requireSelection	a value if a data item must always be selected in the control (Inherited from *SparkListBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	The 0-based index of the selected item, or -1 if no item is selected. (Inherited from *SparkListBase*)
selectedIndices	a list representing the indices of the currently selected item or items. (Inherited from *SparkList*)
selectedItem	The item that is currently selected. (Inherited from *SparkListBase*)

Name	Description
selectedItems	a list of Objects representing the currently selected data items. (Inherited from *SparkList*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useVirtualLayout	the value of the useVirtualLayout property of the layout associated with this control. (Inherited from *SparkListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemsCount	Gets the number of items in the ComboBox. (Inherited from *SparkListBase*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
Open	Dispatched when the user clicks the drop-down button to display the drop-down list. (Inherited from *SparkDropDownListBase*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Dispatched when an item in the list is selected. (Inherited from *SparkListBase*)
SelectIndex	Dispatched when the user clicks on an item in the list or navigates to the item using a keyboard. (Inherited from *SparkList*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkDropDownListBase Class

Description
The SparkDropDownListBase control contains a drop-down list from which the user can select a single value.

Inheritance Hierarchy

- *SparkList*
  - SparkDropDownListBase
    - *SparkComboBox*
    - *SparkDropDownList*

Syntax

```
'Declaration
Public Class SparkDropDownListBase _
Inherits SparkList
```

Properties

Name	Description
allowMultipleSelection	a value if multiple selection is enabled. (Inherited from *SparkList*)
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	the measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableDataContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
caretIndex	The item that is currently in focus. (Inherited from *SparkListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dragEnabled	A flag that indicates whether you can drag items out of this control and drop them on other controls. (Inherited from *SparkList*)
dragMoveEnabled	A flag that indicates whether items can be moved instead of just copied from the control as part of a drag-and-drop operation. (Inherited from *SparkList*)
dropEnabled	A flag that indicates whether dragged items can be dropped onto the control. (Inherited from *SparkList*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)

Name	Description
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
isDropDownOpen	a value whether the drop down is open.
labelField	The name of the field in the data provider items to display as the label. (Inherited from *SparkListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
requireSelection	a value if a data item must always be selected in the control (Inherited from *SparkListBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	The 0-based index of the selected item, or -1 if no item is selected. (Inherited from *SparkListBase*)

Name	Description
selectedIndices	a list representing the indices of the currently selected item or items. (Inherited from *SparkList*)
selectedItem	The item that is currently selected. (Inherited from *SparkListBase*)
selectedItems	a list of Objects representing the currently selected data items. (Inherited from *SparkList*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useVirtualLayout	the value of the useVirtualLayout property of the layout associated with this control. (Inherited from *SparkListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemsCount	Gets the number of items in the ComboBox. (Inherited from *SparkListBase*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)

Name	Description
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
Open	Dispatched when the user clicks the drop-down button to display the drop-down list.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Dispatched when an item in the list is selected. (Inherited from *SparkListBase*)
SelectIndex	Dispatched when the user clicks on an item in the list or navigates to the item using a keyboard. (Inherited from *SparkList*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkForm Class

Description
Identifies a Spark form control.

Inheritance Hierarchy

- *SparkSkinnableContainer*
  - SparkForm

Syntax

```
'Declaration
Public Class SparkForm _
Inherits SparkSkinnableContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)

Name	Description
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkFormItem Class

Description
Identifies a Spark form item control.

Inheritance Hierarchy

- *SparkSkinnableContainer*
  - SparkFormItem

Syntax
```
'Declaration
Public Class SparkFormItem _
Inherits SparkSkinnableContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)

Name	Description
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text of the form items label.
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)

Name	Description
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
required	whether the form item has to have a value to submit the form.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
sequenceLabel	the number of the form item in the form.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkGroup Class

Description
Use this container when you want to manage visual children, both visual components and graphical components.

Inheritance Hierarchy

- *SparkGroupBase*
  - SparkGroup
    - *SparkDataRenderer*
    - *SparkTileGroup*

Syntax
```
'Declaration
Public Class SparkGroup _
Inherits SparkGroupBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	a value whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkGroupBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
blendMode	A value from the BlendMode class that specifies which blend mode to use.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipAndEnableScrolling	a value whether to clip the children to the boundaries of the viewport. (Inherited from *SparkGroupBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkGroupBase*)
contentHeight	The height of the viewport's content. (Inherited from *SparkGroupBase*)
contentWidth	The width of the viewport's contents. (Inherited from *SparkGroupBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)

Name	Description
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkGroupBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalScrollPosition	The x coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component (Inherited from *SparkGroupBase*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkGroupBase*)
luminosityClip	a value whether the luminosity mask clips the masked content. (Inherited from *SparkGroupBase*)
luminosityInvert	a value that controls the calculation of the RGB color value of a graphic element being masked by a luminosity mask. (Inherited from *SparkGroupBase*)
maskType	The mask type. (Inherited from *SparkGroupBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseEnabledWhereTransparent	a value whether the entire bounds of the Group respond to mouse events such as click and roll over. (Inherited from *SparkGroupBase*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkGroupBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkGroupBase*)
numRows	the total number of rows of data available. (Inherited from *SparkGroupBase*)

Name	Description
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizeMode	The ResizeMode for this container. (Inherited from *SparkGroupBase*)
scaleGridBottom	the bottom coordinate of the scale grid.
scaleGridLeft	the left coordinate of the scale grid.
scaleGridRight	the right coordinate of the scale grid.
scaleGridTop	the top coordinate of the scale grid.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalScrollPosition	The y coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component. (Inherited from *SparkGroupBase*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Name	Description
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkGroupBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkGroupBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkGroupBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkGroupBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container. (Inherited from *SparkGroupBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the container's scroll events occur. This event is dispatched if the scrolling is done programmatically or by the user. (Inherited from *SparkGroupBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkGroupBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkGroupBase Class

Description
The GroupBase class defines the base class for components that display visual elements.

Inheritance Hierarchy
- *FlexObject*
  - SparkGroupBase
    - *SparkDataGroup*
    - *SparkGroup*

Syntax
```
'Declaration
Public Class SparkGroupBase _
Inherits FlexObject
```

Properties
Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	a value whether measurement and layout are done when the position or size of a child is changed.
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipAndEnableScrolling	a value whether to clip the children to the boundaries of the viewport.
columnNames	a list containing the names of all columns in the data.
contentHeight	The height of the viewport's content.

Name	Description
contentWidth	The width of the viewport's contents.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalScrollPosition	The x coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child.
luminosityClip	a value whether the luminosity mask clips the masked content.
luminosityInvert	a value that controls the calculation of the RGB color value of a graphic element being masked by a luminosity mask.
maskType	The mask type.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseEnabledWhereTransparent	a value whether the entire bounds of the Group respond to mouse events such as click and roll over.
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numElements	The number of visual elements in this container.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)

Name	Description
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizeMode	The ResizeMode for this container.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalScrollPosition	The y coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/

Name	Description
	Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled.
DragDrop	Dispatched when the dragged item is dropped.
DragStart	Dispatched when the drag/drop operation begins.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)

Name	Description
GetValues	Returns a matrix containing the automation values of all parts of the components.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container.
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the container's scroll events occur. This event is dispatched if the scrolling is done programmatically or by the user.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkImage Class

Description
Identifies a Spark image control.

Inheritance Hierarchy

- *SparkObject*
  - SparkImage

Syntax

```
'Declaration
Public Class SparkImage _
Inherits SparkObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
bytesLoaded	the number of bytes of the image already loaded.
bytesTotal	the total image data in bytes loaded or pending load.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clearOnLoad	whether or not to clear previous image content prior to loading new content.
contentLoaderGrouping	the optional content grouping identifier to pass to the an associated IContentLoader instance's load() method.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
fillMode	a value how the bitmap fills in the dimensions.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalAlign	the horizontal alignment of the content when it does not have a one-to-one aspect ratio and scaleMode is set to BitmapScaleMode.LETTERBOX.

Name	Description
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
preliminaryHeight	an estimate to use for height when the "measured" bounds of the image is requested by layout, but the image data has yet to complete loading.
preliminaryWidth	an estimate to use for width when the "measured" bounds of the image is requested by layout, but the image data has yet to complete loading.
scaleMode	whether the image is scaled when fillMode is set to BitmapFillMode.SCALE.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
smooth	whether to apply a smoothing algorithm to the bitmap image.
source	the content location.
sourceHeight	the unscaled height of the original image data.
sourceWidth	the unscaled width of the original image data.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
trustedSource	a flag denoting whether the currently loaded content is considered loaded from a source whose security policy allows for cross domain image access.
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalAlign	the vertical alignment of the content when it does not have a one-to-one aspect ratio and scaleMode is set to BitmapScaleMode.LETTERBOX.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkLabel Class

Description

Label is a low-level UIComponent that can render one or more lines of uniformly-formatted text.

Inheritance Hierarchy

- *SparkTextBase*
  - SparkLabel

Syntax

```
'Declaration
Public Class SparkLabel _
Inherits SparkTextBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)

Name	Description
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
isTruncated	whether the text has been truncated. (Inherited from *SparkTextBase*)
maxDisplayedLines	An integer which determines whether, and where, the text gets truncated. (Inherited from *SparkTextBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	The text displayed by this text component. (Inherited from *SparkTextBase*)

Name	Description
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled. (Inherited from *SparkTextBase*)
DragDrop	The event triggered when the dragged item is dropped. (Inherited from *SparkTextBase*)
DragStart	The event triggered when the drag/drop operation begins. (Inherited from *SparkTextBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkTextBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout

Name	Description
	parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkList Class

Description
The List control displays a vertical list of items.

Inheritance Hierarchy

- *SparkListBase*
  - SparkList
    - *SparkDropDownListBase*

Syntax

```
'Declaration
Public Class SparkList _
Inherits SparkListBase
```

Properties

Name	Description
allowMultipleSelection	a value if multiple selection is enabled.
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	the measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableDataContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)

Name	Description
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
caretIndex	The item that is currently in focus. (Inherited from *SparkListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
dragEnabled	A flag that indicates whether you can drag items out of this control and drop them on other controls.
dragMoveEnabled	A flag that indicates whether items can be moved instead of just copied from the control as part of a drag-and-drop operation.
dropEnabled	A flag that indicates whether dragged items can be dropped onto the control.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
labelField	The name of the field in the data provider items to display as the label. (Inherited from *SparkListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
requireSelection	a value if a data item must always be selected in the control (Inherited from *SparkListBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	The 0-based index of the selected item, or -1 if no item is selected. (Inherited from *SparkListBase*)
selectedIndices	a list representing the indices of the currently selected item or items.
selectedItem	The item that is currently selected. (Inherited from *SparkListBase*)
selectedItems	a list of Objects representing the currently selected data items.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useVirtualLayout	the value of the useVirtualLayout property of the layout associated with this control. (Inherited from *SparkListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemsCount	Gets the number of items in the ComboBox. (Inherited from *SparkListBase*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Dispatched when an item in the list is selected. (Inherited from *SparkListBase*)
SelectIndex	Dispatched when the user clicks on an item in the list or navigates to the item using a keyboard.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkListBase Class

Description
The ListBase class is the base class for all components that support selection.

Inheritance Hierarchy

- *SparkSkinnableDataContainer*
  - SparkListBase
    - *SparkButtonBar*
    - *SparkList*
    - *SparkTabBar*

Syntax
```
'Declaration
Public Class SparkListBase _
Inherits SparkSkinnableDataContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	the measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableDataContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)

Name	Description
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
caretIndex	The item that is currently in focus.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
labelField	The name of the field in the data provider items to display as the label.
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)

Name	Description
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
requireSelection	a value if a data item must always be selected in the control
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	The 0-based index of the selected item, or -1 if no item is selected.
selectedItem	The item that is currently selected.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useVirtualLayout	the value of the useVirtualLayout property of the layout associated with this control.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemsCount	Gets the number of items in the ComboBox.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Dispatched when an item in the list is selected.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkListLabel Class

Description
A simple rendered object.

Inheritance Hierarchy

- *SparkDataRenderer*
  - SparkListLabel
    - *SparkComplexDisplay*

Syntax

```
'Declaration
Public Class SparkListLabel _
Inherits SparkDataRenderer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	a value whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkGroupBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
automationValue	A value which can be used to uniquely identify this item.
blendMode	A value from the BlendMode class that specifies which blend mode to use. (Inherited from *SparkGroup*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipAndEnableScrolling	a value whether to clip the children to the boundaries of the viewport. (Inherited from *SparkGroupBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkGroupBase*)
contentHeight	The height of the viewport's content. (Inherited from *SparkGroupBase*)
contentWidth	The width of the viewport's contents. (Inherited from *SparkGroupBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)

Name	Description
dragging	a value whether the SparkListLabel is in dragging mode.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkGroupBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalScrollPosition	The x coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component (Inherited from *SparkGroupBase*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the label of the SparkListLabel.
lastVisibleRow	the index of the last visible child. (Inherited from *SparkGroupBase*)
luminosityClip	a value whether the luminosity mask clips the masked content. (Inherited from *SparkGroupBase*)
luminosityInvert	a value that controls the calculation of the RGB color value of a graphic element being masked by a luminosity mask. (Inherited from *SparkGroupBase*)
maskType	The mask type. (Inherited from *SparkGroupBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseEnabledWhereTransparent	a value whether the entire bounds of the Group respond to mouse events such as click and roll over. (Inherited from *SparkGroupBase*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkGroupBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkGroupBase*)

Name	Description
numRows	the total number of rows of data available. (Inherited from *SparkGroupBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizeMode	The ResizeMode for this container. (Inherited from *SparkGroupBase*)
scaleGridBottom	the bottom coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridLeft	the left coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridRight	the right coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridTop	the top coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	a value whether the SparkListLabel is selected.
showsCaret	a value whether the caret is shown.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalScrollPosition	The y coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component. (Inherited from *SparkGroupBase*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkGroupBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkGroupBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkGroupBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkGroupBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container. (Inherited from *SparkGroupBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the container's scroll events occur. This event is dispatched if the scrolling is done programmatically or by the user. (Inherited from *SparkGroupBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkGroupBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkMuteButton Class

Description
The MuteButton class defines the mute button used by the VideoPlayer control.

Inheritance Hierarchy

- *SparkButton*
  - SparkMuteButton

Syntax
```
'Declaration
Public Class SparkMuteButton _
Inherits SparkButton
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
autoRepeat	whether to dispatch repeated buttonDown events if the user holds down the mouse button. (Inherited from *SparkButtonBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
currentState	the current view state of the component. (Inherited from *FlexObject*)
emphasized	whether the default button as requested by the focus manager. (Inherited from *SparkButton*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the ButtonBase control. (Inherited from *SparkButtonBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
muted	a value whether the control is muted.
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
stickyHighlighting	a value whether the button displays its down skin when the user presses it but changes to its over skin when the user drags the mouse off of it. (Inherited from *SparkButtonBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)

Name	Description
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
volume	the volume of the control.
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
MuteChange	Toggles the muting of the control.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkButtonBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkNavigatorContent Class

Description
The NavigatorContent class defines a Spark container that can be used in an MX navigator container, such as the ViewStack, TabNavigator and Accordion containers.

Inheritance Hierarchy

- *SparkSkinnableContainer*
  - SparkNavigatorContent

Syntax

```
'Declaration
Public Class SparkNavigatorContent _
Inherits SparkSkinnableContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)

Name	Description
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)

Name	Description
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkNumericStepper Class

Description
The NumericStepper control lets you select a number from an ordered set. The NumericStepper provides the same functionality as the Spinner component, but adds a TextInput control so that you can directly edit the value of the component, rather than modifying it by using the control's arrow buttons.

Inheritance Hierarchy

- *SparkSpinner*
  - SparkNumericStepper

Syntax

```
'Declaration
Public Class SparkNumericStepper _
Inherits SparkSpinner
```

Properties

Name	Description
allowValueWrap	the behavior of the control for a step when the current value is either the maximum or minimum value. (Inherited from *SparkSpinner*)
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
enableIME	a value whether the IME should be enabled when the component receives focus.
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)

Name	Description
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
imeMode	the IME (Input Method Editor) mode.
maxChars	The maximum number of characters that can be entered in the field.
maximum	The maximum valid value. (Inherited from *SparkRange*)
minimum	The minimum valid value. (Inherited from *SparkRange*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
snapInterval	the snapInterval. (Inherited from *SparkRange*)
stepSize	The amount that the value property changes when the changeValueByStep() method is called. (Inherited from *SparkRange*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	The current value for this range. (Inherited from *SparkRange*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when the value of the NumericStepper changes as a result of user interaction. (Inherited from *SparkSpinner*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Dispatched when the user types, deletes, or pastes text into the control.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SelectText	Dispatched when text is selected.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkSpinner*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkObject Class

Description
The SparkObject class defines the base class for all spark components.

Inheritance Hierarchy

- *FlexObject*
  - SparkObject
    - *SparkButtonBase*
    - *SparkImage*
    - *SparkRange*
    - *SparkRichEditableText*
    - *SparkSkinnableContainerBase*
    - *SparkSkinnableTextBase*
    - *SparkVideoPlayer*

Syntax

```
'Declaration
Public Class SparkObject _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)

Name	Description
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)

Name	Description
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkPanel Class

Description
The Panel class defines a container that includes a title bar, a caption, a border, and a content area for its children.

Inheritance Hierarchy

- *SparkSkinnableContainer*
  - SparkPanel
    - *SparkTitleWindow*

Syntax

```
'Declaration
Public Class SparkPanel _
Inherits SparkSkinnableContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)

Name	Description
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
controlBarVisible	a value whether the control bar is visible.
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)

Name	Description
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
title	the title or caption displayed in the title bar.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkPopUpAnchor Class

Description
Identifies a Spark PopUpAnchor control.

Inheritance Hierarchy

- *FlexObject*
  - SparkPopUpAnchor

Syntax

```
'Declaration
Public Class SparkPopUpAnchor _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
displayPopUp	whether the popUp is visible.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
popUpHeightMatchesAnchorHeight	whether the height of the popUp control is set to the value of the PopUpAnchor's height.
popUpPosition	the position of the popUp control when it is opened, relative to the PopUpAnchor component.
popUpWidthMatchesAnchorWidth	whether the width of the popUp control is set to the value of the PopUpAnchor's width.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkRadioButton Class

Description
The RadioButton component allows the user make a single choice within a set of mutually exclusive choices. A RadioButtonGroup is composed of two or more RadioButton components with the same groupName property.

Inheritance Hierarchy

- *SparkToggleButtonBase*
  - SparkRadioButton

Syntax

```
'Declaration
Public Class SparkRadioButton _
Inherits SparkToggleButtonBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
autoRepeat	whether to dispatch repeated buttonDown events if the user holds down the mouse button. (Inherited from *SparkButtonBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
groupName	the name of the group to which this RadioButton component belongs, or specifies the value of the id property of a RadioButtonGroup component if this RadioButton is part of a group defined by a RadioButtonGroup component.
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the ButtonBase control. (Inherited from *SparkButtonBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list,

Name	Description
	use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	whether the button is in the down state, and false if it is in the up state. (Inherited from *SparkToggleButtonBase*)
stickyHighlighting	a value whether the button displays its down skin when the user presses it but changes to its over skin when the user drags the mouse off of it. (Inherited from *SparkButtonBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Name	Description
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled.
DragDrop	The event triggered when the dragged item is dropped.
DragStart	The event triggered when the drag/drop operation begins.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkButtonBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkRange Class

Description
The Range class holds a value and an allowed range for that value, defined by minimum and maximum properties.

Inheritance Hierarchy

- *SparkObject*
  - SparkRange
    - *SparkSpinner*
    - *SparkTrackBase*

Syntax

```
'Declaration
Public Class SparkRange _
Inherits SparkObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)

Name	Description
maximum	The maximum valid value.
minimum	The minimum valid value.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
snapInterval	the snapInterval.
stepSize	The amount that the value property changes when the changeValueByStep() method is called.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	The current value for this range.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)

Name	Description
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkRichEditableText Class

Description
RichEditableText is a low-level UIComponent for displaying, scrolling, selecting, and editing richly-formatted text.

Inheritance Hierarchy

- *SparkObject*
  - SparkRichEditableText

Syntax

```
'Declaration
Public Class SparkRichEditableText _
Inherits SparkObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipAndEnableScrolling	whether to clip the children to the boundaries of the viewport.
columnNames	a list containing the names of all columns in the data.
contentHeight	The height of the text.
contentWidth	The width of the text.
currentState	the current view state of the component. (Inherited from *FlexObject*)
displayAsPassword	whether the text field is a password text field.
editable	A flag indicating whether the user is allowed to edit the text in this control.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
enableIME	A flag that indicates whether the IME should be enabled when the component receives focus.
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)

Name	Description
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
heightInLines	The default height of the control, measured in lines.
horizontalScrollPosition	The number of pixels by which the text is scrolled horizontally.
id	the ID of the component. (Inherited from *FlexObject*)
imeMode	the IME (input method editor) mode.
lastVisibleRow	the index of the last visible child.
maxChars	The maximum number of characters that the text field can contain, as entered by a user.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
multiline	whether the user can enter multiline text.
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
restrict	the set of characters that a user can enter into the text field.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	A flag indicating whether the content is selectable with the mouse, or with the keyboard when the control has the keyboard focus.

Name	Description
selectionActivePosition	A character position, relative to the beginning of the text String, specifying the end of the selection that moves when the selection is extended with the arrow keys.
selectionAnchorPosition	A character position, relative to the beginning of the text String, specifying the end of the selection that stays fixed when the selection is extended with the arrow keys.
selectionHighlighting	a value whether the text selection is highlighted.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	The text String displayed by this component.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalScrollPosition	The number of pixels by which the text is scrolled vertically.
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
widthInChars	The default width of the control, measured in em units.
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled.
DragDrop	The event triggered when the dragged item is dropped.
DragStart	The event triggered when the drag/drop operation begins.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.

Name	Description
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Dispatched when the user types, deletes, or pastes text into the control.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container.
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SelectText	Dispatched when text is selected.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkRichText Class

Description
RichText is a low-level UIComponent that can display one or more lines of richly-formatted text and embedded images.

Inheritance Hierarchy

- *SparkTextBase*
  - SparkRichText

Syntax

```
'Declaration
Public Class SparkRichText _
Inherits SparkTextBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
isTruncated	whether the text has been truncated. (Inherited from *SparkTextBase*)
luminosityClip	whether the luminosity mask clips the masked content.
luminosityInvert	whether the calculation of the RGB color value of a graphic element is masked by a luminosity mask.
maskType	how the mask is applied to the component. The possible values are MaskType.CLIP, MaskType.ALPHA and MaskType.LUMINOSITY. Clip Masking When masking in clip mode, a clipping masks is reduced to 1-bit.

Name	Description
maxDisplayedLines	An integer which determines whether, and where, the text gets truncated. (Inherited from *SparkTextBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	The text displayed by this text component. (Inherited from *SparkTextBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)

Name	Description
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled. (Inherited from *SparkTextBase*)
DragDrop	The event triggered when the dragged item is dropped. (Inherited from *SparkTextBase*)
DragStart	The event triggered when the drag/drop operation begins. (Inherited from *SparkTextBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkTextBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkScrollBar Class

Description
The ScrollBarBase class helps to position the portion of data that is displayed when there is too much data to fit in a display area.

Inheritance Hierarchy

- *SparkTrackBase*
  - SparkScrollBar

Syntax

```
'Declaration
Public Class SparkScrollBar _
Inherits SparkTrackBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
maximum	The maximum valid value. (Inherited from *SparkRange*)
minimum	The minimum valid value. (Inherited from *SparkRange*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
pageSize	The change in the value of the value property when you call the changeValueByPage() method.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
snapInterval	the snapInterval. (Inherited from *SparkRange*)
stepSize	The amount that the value property changes when the changeValueByStep() method is called. (Inherited from *SparkRange*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	The current value for this range. (Inherited from *SparkRange*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when the value of the NumericStepper changes as a result of user interaction. (Inherited from *SparkTrackBase*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkTrackBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use

Name	Description
	WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkSkinnableContainer Class

Description
The SparkSkinnableContainer class is the base class for skinnable containers that have visual content.

Inheritance Hierarchy

- *SparkSkinnableContainerBase*
  - SparkSkinnableContainer
    - *SparkApplication*
    - *SparkBorderContainer*
    - *SparkForm*
    - *SparkFormItem*
    - *SparkNavigatorContent*
    - *SparkPanel*
    - *SparkSkinnablePopUpContainer*
    - *SparkWindow*

Syntax

```
'Declaration
Public Class SparkSkinnableContainer _
Inherits SparkSkinnableContainerBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	whether measurement and layout are done when the position or size of a child is changed.
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)

Name	Description
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
creationPolicy	the content creation policy for this component.
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numElements	The number of visual elements in this container.
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)

Name	Description
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha

Name	Description
	channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)

Name	Description
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkSkinnableContainerBase Class

Description
Base class for spark container components.

Inheritance Hierarchy

- *SparkObject*
  - SparkSkinnableContainerBase

- *SparkDataGrid*
- *SparkSkinnableContainer*
- *SparkSkinnableDataContainer*

Syntax

```
'Declaration
Public Class SparkSkinnableContainerBase _
Inherits SparkObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data.
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list,

Name	Description
	use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled.
DragDrop	Dispatched when the dragged item is dropped.
DragStart	Dispatched when the drag/drop operation begins.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkSkinnableDataContainer Class

Description
The SkinnableDataContainer class is the base container class for data items.

Inheritance Hierarchy

- *SparkSkinnableContainerBase*

- SparkSkinnableDataContainer
  - *SparkListBase*

Syntax

```
'Declaration
Public Class SparkSkinnableDataContainer _
Inherits SparkSkinnableContainerBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	the measurement and layout are done when the position or size of a child is changed.
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that

Name	Description
	is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)

Name	Description
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkSkinnablePopUpContainer Class

Description
Identifies a SparkSkinnablePopUpContainer control.

Inheritance Hierarchy

- *SparkSkinnableContainer*
  - SparkSkinnablePopUpContainer

Syntax

```
'Declaration
Public Class SparkSkinnablePopUpContainer _
Inherits SparkSkinnableContainer
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)

Name	Description
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
isOpen	whether the container is open.
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)

Name	Description
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkSkinnableTextBase Class

Description

The base class for skinnable components, such as the Spark TextInput and TextArea, that include an instance of RichEditableText in their skin to provide rich text display, scrolling, selection, and editing.

Inheritance Hierarchy

- *SparkObject*
  - SparkSkinnableTextBase
    - *SparkTextArea*
    - *SparkTextInput*

Syntax

```
'Declaration
Public Class SparkSkinnableTextBase _
Inherits SparkObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data.
currentState	the current view state of the component. (Inherited from *FlexObject*)
displayAsPassword	whether the text field is a password text field.
editable	A flag indicating whether the user is allowed to edit the text in this control.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
enableIME	A flag that indicates whether the IME should be enabled when the component receives focus.
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child.
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
imeMode	the IME (input method editor) mode.
lastVisibleRow	the index of the last visible child.
maxChars	The maximum number of characters that the text field can contain, as entered by a user.
maxWidth	The maximum recommended width of the component to be considered by the parent during layout.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of

Name	Description
	this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available.
numRows	the total number of rows of data available.
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
restrict	the set of characters that a user can enter into the text field.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	A flag indicating whether the content is selectable with the mouse, or with the keyboard when the control has the keyboard focus.
selectionActivePosition	A character position, relative to the beginning of the text String, specifying the end of the selection that moves when the selection is extended with the arrow keys.
selectionAnchorPosition	A character position, relative to the beginning of the text String, specifying the end of the selection that stays fixed when the selection is extended with the arrow keys.
selectionHighlighting	a value whether the text selection is highlighted.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	The text String displayed by this component.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Dispatched when the user types, deletes, or pastes text into the control.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container.
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SelectText	Dispatched when text is selected.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkSlider Class

Description

Lets users select a value by moving a slider thumb between the end points of the slider track.

Inheritance Hierarchy

- *SparkTrackBase*
  - SparkSlider
    - *SparkVolumeBar*

Syntax

```
'Declaration
Public Class SparkSlider _
Inherits SparkTrackBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)

Name	Description
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
maximum	The maximum valid value. (Inherited from *SparkRange*)
minimum	The minimum valid value. (Inherited from *SparkRange*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
snapInterval	the snapInterval. (Inherited from *SparkRange*)
stepSize	The amount that the value property changes when the changeValueByStep() method is called. (Inherited from *SparkRange*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)

Name	Description
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	The current value for this range. (Inherited from *SparkRange*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when the value of the NumericStepper changes as a result of user interaction. (Inherited from *SparkTrackBase*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkTrackBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout

Name	Description
	parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkSpinner Class

Description
A Spinner component selects a value from an ordered set. It uses two buttons that increase or decrease the current value based on the current value of the stepSize property.

Inheritance Hierarchy

- *SparkRange*
  - SparkSpinner
    - *SparkNumericStepper*

Syntax

```
'Declaration
Public Class SparkSpinner _
Inherits SparkRange
```

Properties

Name	Description
allowValueWrap	the behavior of the control for a step when the current value is either the maximum or minimum value.
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)

Name	Description
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
maximum	The maximum valid value. (Inherited from *SparkRange*)
minimum	The minimum valid value. (Inherited from *SparkRange*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
snapInterval	the snapInterval. (Inherited from *SparkRange*)

Name	Description
stepSize	The amount that the value property changes when the changeValueByStep() method is called. (Inherited from *SparkRange*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	The current value for this range. (Inherited from *SparkRange*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when the value of the NumericStepper changes as a result of user interaction.

Name	Description
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a

Name	Description
	timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkTabBar Class

Description

The TabBar class displays a set of identical tabs. One tab can be selected at a time, and the first tab is selected by default.

Inheritance Hierarchy

- *SparkListBase*
  - SparkTabBar

Syntax

```
'Declaration
Public Class SparkTabBar _
Inherits SparkListBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)

Name	Description
autoLayout	the measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableDataContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
caretIndex	The item that is currently in focus. (Inherited from *SparkListBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
labelField	The name of the field in the data provider items to display as the label. (Inherited from *SparkListBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
requireSelection	a value if a data item must always be selected in the control (Inherited from *SparkListBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectedIndex	The 0-based index of the selected item, or -1 if no item is selected. (Inherited from *SparkListBase*)
selectedItem	The item that is currently selected. (Inherited from *SparkListBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
useVirtualLayout	the value of the useVirtualLayout property of the layout associated with this control. (Inherited from *SparkListBase*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)

Name	Description
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemsCount	Gets the number of items in the ComboBox. (Inherited from *SparkListBase*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Dispatched when an item in the list is selected. (Inherited from *SparkListBase*)
SelectIndex	Dispatched when the user clicks on an item in the list or navigates to the item using a keyboard.
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkListBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkTextArea Class

Description

TextArea is a text-entry control that lets users enter and edit multiple lines of richly formatted text. It can display horizontal and vertical scrollbars for scrolling through the text and supports vertical scrolling with the mouse wheel.

Inheritance Hierarchy

- *SparkSkinnableTextBase*
  - SparkTextArea

Syntax

```
'Declaration
Public Class SparkTextArea _
Inherits SparkSkinnableTextBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableTextBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
displayAsPassword	whether the text field is a password text field. (Inherited from *SparkSkinnableTextBase*)
editable	A flag indicating whether the user is allowed to edit the text in this control. (Inherited from *SparkSkinnableTextBase*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
enableIME	A flag that indicates whether the IME should be enabled when the component receives focus. (Inherited from *SparkSkinnableTextBase*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableTextBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
heightInLines	The default height of the control, measured in lines.
id	the ID of the component. (Inherited from *FlexObject*)
imeMode	the IME (input method editor) mode. (Inherited from *SparkSkinnableTextBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableTextBase*)
maxChars	The maximum number of characters that the text field can contain, as entered by a user. (Inherited from *SparkSkinnableTextBase*)
maxWidth	The maximum recommended width of the component to be considered by the parent during layout. (Inherited from *SparkSkinnableTextBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableTextBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableTextBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
restrict	the set of characters that a user can enter into the text field. (Inherited from *SparkSkinnableTextBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	A flag indicating whether the content is selectable with the mouse, or with the keyboard when the control has the keyboard focus. (Inherited from *SparkSkinnableTextBase*)
selectionActivePosition	A character position, relative to the beginning of the text String, specifying the end of the selection that moves when the selection is extended with the arrow keys. (Inherited from *SparkSkinnableTextBase*)
selectionAnchorPosition	A character position, relative to the beginning of the text String, specifying the end of the selection that stays fixed when the selection is extended with the arrow keys. (Inherited from *SparkSkinnableTextBase*)
selectionHighlighting	a value whether the text selection is highlighted. (Inherited from *SparkSkinnableTextBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	The text String displayed by this component. (Inherited from *SparkSkinnableTextBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)

Name	Description
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
widthInChars	The default width of the control, measured in em units.
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled.
DragDrop	The event triggered when the dragged item is dropped.
DragStart	The event triggered when the drag/drop operation begins.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableTextBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Input	Dispatched when the user types, deletes, or pastes text into the control. (Inherited from *SparkSkinnableTextBase*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container. (Inherited from *SparkSkinnableTextBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SelectText	Dispatched when text is selected. (Inherited from *SparkSkinnableTextBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkSkinnableTextBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkTextBase Class

Description

The base class for Spark text controls such as Label and RichText which display text using CSS styles for the default format.

Inheritance Hierarchy

- *FlexObject*
  - SparkTextBase
    - *SparkLabel*
    - *SparkRichText*

Syntax

```
'Declaration
Public Class SparkTextBase _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)

Name	Description
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
isTruncated	whether the text has been truncated.
maxDisplayedLines	An integer which determines whether, and where, the text gets truncated.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)

Name	Description
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
text	The text displayed by this text component.
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled.
DragDrop	The event triggered when the dragged item is dropped.
DragStart	The event triggered when the drag/drop operation begins.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkTextInput Class

Description
TextInput is a text-entry control that lets users enter and edit a single line of uniformly-formatted text.

Inheritance Hierarchy
- *SparkSkinnableTextBase*
  - SparkTextInput

Syntax
```
'Declaration
Public Class SparkTextInput _
Inherits SparkSkinnableTextBase
```

Properties
Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)

Name	Description
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableTextBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
displayAsPassword	whether the text field is a password text field. (Inherited from *SparkSkinnableTextBase*)
editable	A flag indicating whether the user is allowed to edit the text in this control. (Inherited from *SparkSkinnableTextBase*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
enableIME	A flag that indicates whether the IME should be enabled when the component receives focus. (Inherited from *SparkSkinnableTextBase*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableTextBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
imeMode	the IME (input method editor) mode. (Inherited from *SparkSkinnableTextBase*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableTextBase*)
maxChars	The maximum number of characters that the text field can contain, as entered by a user. (Inherited from *SparkSkinnableTextBase*)
maxWidth	The maximum recommended width of the component to be considered by the parent during layout. (Inherited from *SparkSkinnableTextBase*)

Name	Description
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableTextBase*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableTextBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
restrict	the set of characters that a user can enter into the text field. (Inherited from *SparkSkinnableTextBase*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selectable	A flag indicating whether the content is selectable with the mouse, or with the keyboard when the control has the keyboard focus. (Inherited from *SparkSkinnableTextBase*)
selectionActivePosition	A character position, relative to the beginning of the text String, specifying the end of the selection that moves when the selection is extended with the arrow keys. (Inherited from *SparkSkinnableTextBase*)
selectionAnchorPosition	A character position, relative to the beginning of the text String, specifying the end of the selection that stays fixed when the selection is extended with the arrow keys. (Inherited from *SparkSkinnableTextBase*)
selectionHighlighting	a value whether the text selection is highlighted. (Inherited from *SparkSkinnableTextBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
text	The text String displayed by this component. (Inherited from *SparkSkinnableTextBase*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
widthInChars	The default width of the control, measured in em units.
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled.
DragDrop	The event triggered when the dragged item is dropped.
DragStart	The event triggered when the drag/drop operation begins.

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableTextBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Input	Dispatched when the user types, deletes, or pastes text into the control. (Inherited from *SparkSkinnableTextBase*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container. (Inherited from *SparkSkinnableTextBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SelectText	Dispatched when text is selected. (Inherited from *SparkSkinnableTextBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkSkinnableTextBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkTileGroup Class

Description
The TileGroup container is an instance of the Group container that uses the TileLayout class.

Inheritance Hierarchy
- *SparkGroup*
  - SparkTileGroup

Syntax
```
'Declaration
Public Class SparkTileGroup _
Inherits SparkGroup
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	a value whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkGroupBase*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
blendMode	A value from the BlendMode class that specifies which blend mode to use. (Inherited from *SparkGroup*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
clipAndEnableScrolling	a value whether to clip the children to the boundaries of the viewport. (Inherited from *SparkGroupBase*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkGroupBase*)
contentHeight	The height of the viewport's content. (Inherited from *SparkGroupBase*)
contentWidth	The width of the viewport's contents. (Inherited from *SparkGroupBase*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkGroupBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
horizontalScrollPosition	The x coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component (Inherited from *SparkGroupBase*)
id	the ID of the component. (Inherited from *FlexObject*)

Name	Description
lastVisibleRow	the index of the last visible child. (Inherited from *SparkGroupBase*)
luminosityClip	a value whether the luminosity mask clips the masked content. (Inherited from *SparkGroupBase*)
luminosityInvert	a value that controls the calculation of the RGB color value of a graphic element being masked by a luminosity mask. (Inherited from *SparkGroupBase*)
maskType	The mask type. (Inherited from *SparkGroupBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
mouseEnabledWhereTransparent	a value whether the entire bounds of the Group respond to mouse events such as click and roll over. (Inherited from *SparkGroupBase*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkGroupBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkGroupBase*)
numRows	the total number of rows of data available. (Inherited from *SparkGroupBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizeMode	The ResizeMode for this container. (Inherited from *SparkGroupBase*)
scaleGridBottom	the bottom coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridLeft	the left coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridRight	the right coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleGridTop	the top coordinate of the scale grid. (Inherited from *SparkGroup*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)

Name	Description
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
verticalScrollPosition	The y coordinate of the origin of the viewport in the component's coordinate system, where the default value is (0,0) corresponding to the upper-left corner of the component. (Inherited from *SparkGroupBase*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkGroupBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkGroupBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkGroupBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkGroupBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MouseScroll	Dispatched when the mouse is used to move the scroll bars on this container. (Inherited from *SparkGroupBase*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Dispatched when the container's scroll events occur. This event is dispatched if the scrolling is done programmatically or by the user. (Inherited from *SparkGroupBase*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkGroupBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkTitleWindow Class

Description
The TitleWindow class extends Panel to include a close button and move area.

Inheritance Hierarchy

- *SparkPanel*
  - SparkTitleWindow

Syntax

```
'Declaration
Public Class SparkTitleWindow _
Inherits SparkPanel
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
controlBarVisible	a value whether the control bar is visible. (Inherited from *SparkPanel*)
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)

Name	Description
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
title	the title or caption displayed in the title bar. (Inherited from *SparkPanel*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkPanel*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkToggleButton Class

Description
The ToggleButton component defines a toggle button. Clicking the button toggles it between the up and an down states.

Inheritance Hierarchy

- *SparkToggleButtonBase*
  - SparkToggleButton
    - *SparkButtonBarButton*

Syntax

```
'Declaration
Public Class SparkToggleButton _
Inherits SparkToggleButtonBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
autoRepeat	whether to dispatch repeated buttonDown events if the user holds down the mouse button. (Inherited from *SparkButtonBase*)

Name	Description
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the ButtonBase control. (Inherited from *SparkButtonBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	whether the button is in the down state, and false if it is in the up state. (Inherited from *SparkToggleButtonBase*)
stickyHighlighting	a value whether the button displays its down skin when the user presses it but changes to its over skin when the user drags the mouse off of it. (Inherited from *SparkButtonBase*)

Name	Description
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	The event triggered when a drag operation is cancelled.
DragDrop	The event triggered when the dragged item is dropped.

Name	Description
DragStart	The event triggered when the drag/drop operation begins.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkButtonBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkToggleButtonBase Class

Description

The ToggleButtonBase component is the base class for the Spark button components that support the selected property.

Inheritance Hierarchy

- *SparkButtonBase*
  - SparkToggleButtonBase
    - *SparkCheckBox*
    - *SparkRadioButton*
    - *SparkToggleButton*

Syntax

```
'Declaration
Public Class SparkToggleButtonBase _
Inherits SparkButtonBase
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
autoRepeat	whether to dispatch repeated buttonDown events if the user holds down the mouse button. (Inherited from *SparkButtonBase*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
label	the text to appear on the ButtonBase control. (Inherited from *SparkButtonBase*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is

Name	Description
	NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
selected	whether the button is in the down state, and false if it is in the up state.
stickyHighlighting	a value whether the button displays its down skin when the user presses it but changes to its over skin when the user drags the mouse off of it. (Inherited from *SparkButtonBase*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha

Name	Description
	channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkButtonBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkTrackBase Class

Description
The TrackBase class is a base class for components with a track and one or more thumb buttons, such as Slider and ScrollBar.

Inheritance Hierarchy

- *SparkRange*
  - SparkTrackBase
    - *SparkScrollBar*
    - *SparkSlider*

Syntax

```
'Declaration
Public Class SparkTrackBase _
Inherits SparkRange
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
maximum	The maximum valid value. (Inherited from *SparkRange*)
minimum	The minimum valid value. (Inherited from *SparkRange*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)

Name	Description
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
snapInterval	the snapInterval. (Inherited from *SparkRange*)
stepSize	The amount that the value property changes when the changeValueByStep() method is called. (Inherited from *SparkRange*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	The current value for this range. (Inherited from *SparkRange*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when the value of the NumericStepper changes as a result of user interaction.
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkVideoDisplay Class

Description
Identifies a Spark video display control.

Inheritance Hierarchy

- *FlexObject*

- SparkVideoDisplay

Syntax

```
'Declaration
Public Class SparkVideoDisplay _
Inherits FlexObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
autoDisplayFirstFrame	whether the video is loaded when the source is set.
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
autoPlay	whether the video starts playing immediately when the source property is set.
autoRewind	whether the FLV file should rewind to the first frame when play stops.
bytesLoaded	the number of bytes of data that have been downloaded into the application.
bytesTotal	the total size in bytes of the data being downloaded into the application.
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
currentTime	the current time of the playhead, measured in seconds, since the video starting playing.
duration	the duration of the video's playback, in seconds.
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)

Name	Description
loop	whether the media should play again after playback has completed.
mediaPlayerState	the current state of the video.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
muted	whether the video is muted.
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
pauseWhenHidden	whether the video continues to play when it is "hidden".
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
playing	a value whether the video is playing or is attempting to play.
scaleMode	the different ways of sizing the video content.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
volume	the volume level, specified as a value between 0 and 1.
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkVideoPlayer Class

Description
The VideoPlayer control is a skinnable video player that supports progressive download, multi-bitrate streaming, and streaming video.

Inheritance Hierarchy

- *SparkObject*
  - SparkVideoPlayer

Syntax

```
'Declaration
Public Class SparkVideoPlayer _
Inherits SparkObject
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)

Name	Description
id	the ID of the component. (Inherited from *FlexObject*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
muted	a value whether the control is muted.
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
playing	a value if the video is playing or is attempting to play.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
volume	The volume level, specified as a value between 0 and 1.
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)

Name	Description
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkVolumeBar Class

Description
The VolumeBar class defines a drop-down slider to control the volume of the VideoDisplay control.

Inheritance Hierarchy

- *SparkSlider*
  - SparkVolumeBar

Syntax
```
'Declaration
Public Class SparkVolumeBar _
Inherits SparkSlider
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
isDropDownOpen	a value whether the volumebar is open.
maximum	The maximum valid value. (Inherited from *SparkRange*)

Name	Description
minimum	The minimum valid value. (Inherited from *SparkRange*)
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
muted	a value whether the control is muted.
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
snapInterval	the snapInterval. (Inherited from *SparkRange*)
stepSize	The amount that the value property changes when the changeValueByStep() method is called. (Inherited from *SparkRange*)
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
value	The current value for this range. (Inherited from *SparkRange*)

Name	Description
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Change	Dispatched when the value of the NumericStepper changes as a result of user interaction. (Inherited from *SparkTrackBase*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *FlexObject*)
MuteChange	Dispatched when the mute status changes as a result of user interaction.
Open	Dispatched when the user clicks the drop-down button to display the drop-down list.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkTrackBase*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a

Name	Description
	WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkWindow Class

Description

The Window is a top-level container for additional windows in an AIR desktop application.

Inheritance Hierarchy

- *SparkSkinnableContainer*
  - SparkWindow

Syntax

```
'Declaration
Public Class SparkWindow _
Inherits SparkSkinnableContainer _
Implements IMoveable
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alwaysInFront	a value whether the underlying NativeWindow is always in front of other windows.
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
closed	a value whether the underlying window has been closed.

Name	Description
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
isAIRWindow	a value whether this window is an AIR window.
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
maxHeight	The maximum recommended height of the component to be considered by the parent during layout.
maximizable	a value whether the window can be maximized.
maxWidth	The maximum recommended width of the component to be considered by the parent during layout.
minHeight	The minimum recommended height of the component to be considered by the parent during layout.
minimizable	a value whether the window can be minimized.
minWidth	The minimum recommended width of the component to be considered by the parent during layout.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)

Name	Description
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizable	a value whether the window can be resized.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
showStatusBar	a value whether the status bar is visible.
status	The string that appears in the status bar, if it is visible.
systemChrome	the type of system chrome (if any) the window has.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
title	The title that appears in the window title bar and the taskbar.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
transparent	a value whether the window is transparent.
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)

Name	Description
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *IMoveable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
StateChange	Dispatched when the displaystatus of the window is changed.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SparkWindowedApplication Class

Description
The WindowedApplication defines the application container that you use to create Flex applications for AIR applications.

Inheritance Hierarchy

- *SparkApplication*
  - SparkWindowedApplication

Syntax

```
'Declaration
Public Class SparkWindowedApplication _
Inherits SparkApplication _
Implements IMoveable
```

Properties

Name	Description
alpha	the alpha transparency value of the object specified. (Inherited from *FlexDisplayObject*)
alwaysInFront	a value whether the underlying NativeWindow is always in front of other windows.
applicationID	The identifier that AIR uses to identify the application.
autoExit	a value whether the AIR application will quit when the last window closes or will continue running in the background.
autoLayout	whether measurement and layout are done when the position or size of a child is changed. (Inherited from *SparkSkinnableContainer*)
automationClassName	a string containing the automation class name for the element. (Inherited from *FlexObject*)
automationIndex	a string containing the automation index for the element. (Inherited from *FlexObject*)
automationName	the name that can be used as an identifier for this object. (Inherited from *FlexObject*)
className	the name of this instance's class, such as *Button*. (Inherited from *FlexObject*)
closed	a value whether the underlying window has been closed.
colorCorrection	The value of the stage's colorCorrection property. (Inherited from *SparkApplication*)
columnNames	a list containing the names of all columns in the data. (Inherited from *SparkSkinnableContainerBase*)
controlBarVisible	whether the control bar is visible. (Inherited from *SparkApplication*)
creationPolicy	the content creation policy for this component. (Inherited from *SparkSkinnableContainer*)
currentState	the current view state of the component. (Inherited from *FlexObject*)
deferredContentCreated	whether deferred content has been created. (Inherited from *SparkSkinnableContainer*)
enabled	whether the component can accept user interaction. (Inherited from *FlexObject*)
errorColor	the color of the component highlight when validation fails. (Inherited from *FlexObject*)
errorString	the text that will be displayed by a component's error tip when a component is monitored and validation fails. (Inherited from *FlexObject*)
firstVisibleRow	the index of the first visible child. (Inherited from *SparkSkinnableContainerBase*)

Name	Description
focusEnabled	whether the component can receive focus when tabbed to. (Inherited from *FlexObject*)
height	the height of the display object, in pixels. (Inherited from *FlexDisplayObject*)
id	the ID of the component. (Inherited from *FlexObject*)
lastVisibleRow	the index of the last visible child. (Inherited from *SparkSkinnableContainerBase*)
maxHeight	The maximum recommended height of the component to be considered by the parent during layout.
maximizable	a value whether the window can be maximized.
maxWidth	The maximum recommended width of the component to be considered by the parent during layout.
minHeight	The minimum recommended height of the component to be considered by the parent during layout.
minimizable	a value whether the window can be minimized.
minWidth	The minimum recommended width of the component to be considered by the parent during layout.
mouseEnabled	whether this object receives mouse messages. The default value is true, which means that by default any InteractiveObject instance that is on the display list receives mouse events. If mouseEnabled is set to false, the instance does not receive any mouse events. Any children of this instance on the display list are not affected. To change the mouseEnabled behavior for all children of an object on the display list, use flash.display.DisplayObjectContainer.mouseChildren. (Inherited from *FlexDisplayObject*)
numAutomationChildren	the number of automation children this container has. (Inherited from *FlexObject*)
numColumns	the total number of columns in the data available. (Inherited from *SparkSkinnableContainerBase*)
numElements	The number of visual elements in this container. (Inherited from *SparkSkinnableContainer*)
numRows	the total number of rows of data available. (Inherited from *SparkSkinnableContainerBase*)
percentHeight	the number that specifies the height of a component as a percentage of its parent's size. (Inherited from *FlexObject*)
percentWidth	the number that specifies the width of a component as a percentage of its parent's size. Allowed values are 0-100. The default value is NaN. Setting the width or explicitWidth properties resets this property to NaN. (Inherited from *FlexObject*)
resizable	a value whether the window can be resized.
scaleX	the number that specifies the horizontal scaling factor. (Inherited from *FlexObject*)

Name	Description
scaleY	the number that specifies the vertical scaling factor. (Inherited from *FlexObject*)
showStatusBar	a value whether the status bar is visible.
status	The string that appears in the status bar, if it is visible.
systemChrome	the type of system chrome (if any) the window has.
tabChildren	whether the children of the display object are tab enabled. The default is true. (Inherited from *FlexDisplayObject*)
tabEnabled	whether this object is in the tab order. (Inherited from *FlexDisplayObject*)
tabIndex	the tab ordering of objects in a SWF file. (Inherited from *FlexDisplayObject*)
Text	The text of the control. (Inherited from *TestObject*)
themeColor	the theme color of a component. (Inherited from *FlexObject*)
title	The title that appears in the window title bar and the taskbar.
toolTip	the text to display in the ToolTip. The default value is `Nothing`. (Inherited from *FlexObject*)
transparent	a value whether the window is transparent.
url	The URL from which this Application's SWF file was loaded. (Inherited from *SparkApplication*)
useHandCursor	whether the pointing hand cursor appears when the mouse rolls over an element. The default value is true. If useHandCursor is false, the arrow pointer is used instead. (Inherited from *FlexObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
viewSourceURL	the URL where the application's source can be viewed. (Inherited from *SparkApplication*)
visible	whether the display object is visible. (Inherited from *FlexDisplayObject*)
width	the width of the display object, in pixels. (Inherited from *FlexDisplayObject*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)
x	the x coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)
y	the y coordinate of the DisplayObject instance relative to the local coordinates of the parent DisplayObjectContainer. (Inherited from *FlexDisplayObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeFocus	Changes the current focus. (Inherited from *FlexObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DragCancel	Dispatched when a drag operation is cancelled. (Inherited from *SparkSkinnableContainerBase*)
DragDrop	Dispatched when the dragged item is dropped. (Inherited from *SparkSkinnableContainerBase*)
DragStart	Dispatched when the drag/drop operation begins. (Inherited from *SparkSkinnableContainerBase*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)

Name	Description
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyle	Returns the style value for the given style. For available styles consider the Apache Flex sdk documentation. (Inherited from *FlexDisplayObject*)
GetValues	Returns a matrix containing the automation values of all parts of the components. (Inherited from *SparkSkinnableContainerBase*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Move	Moves the component to a specified position within its parent. Calling this method is the same as setting the component's x and y properties. (Inherited from *IMoveable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Sets the focus to this component. The component may in turn pass focus to a subcomponent. (Inherited from *FlexObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ToString	Returns a string representation of the control. (Inherited from *FlexDisplayObject*)
Type	Dispatched when the user presses a key. (Inherited from *SparkApplication*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Java SWT Class Reference

Lists the available classes for testing Java SWT controls.

CBanner Class

Description

The class used to layout the toolbar area and perspective switching toolbar in the workbench.

Inheritance Hierarchy

- *Control*
  - CBanner

Syntax

```
'Declaration
Public Class CBanner _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Bottom	The control located at the bottom of the banner.

Name	Description
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
Left	The control located on the left side of the banner.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Right	The control located on the right side of the banner.
RightMinimumSize	The minimum size of the control located on the right side of the banner.
RightWidth	The width of the control located on the right side of the banner.
Simple	Whether the banner is rendered with a simple, traditional shape.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

CoolBar Class

Description
A container class that contains SWT widgets, primarily toolbars.

Inheritance Hierarchy

- *Control*
  - CoolBar

Syntax
```
'Declaration
Public Class CoolBar _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the coolbar.
Items	A list of items in the coolbar.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this

Name	Description
	property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

CoolItem Class

Description
Contains selectable user interface objects that represent the areas of a CoolBar that can be dynamically positioned.

Inheritance Hierarchy

- *Item*
  - CoolItem

Syntax

```
'Declaration
Public Class CoolItem _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Content	The content of the areas that can be dynamically positioned.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

CTabFolder Class

Description
Class for an enhanced tab folder, typically having decorated tabs and the minimize, maximize, and close buttons.

Inheritance Hierarchy

- *Control*
  - CTabFolder

Syntax
```
'Declaration
Public Class CTabFolder _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the tab folder.
Items	A list of items in the tab folder.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The index of the selected item.
SelectedItem	The selected tab item.
SelectedItemText	The text of the selected item or an empty string if no item is selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the visible page.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Resizes the CTabFolder to its full (maximized) size.
Minimize	Resizes the CTabFolder to the minimum size allowed.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the CTabFolder to its previous size.
Select	Selects an item from the CTabFolder.

Name	Description
SelectList	Selects one or more items in the tab.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

CTabItem Class

Description
The class for an enhanced tab within a CTabFolder.

Inheritance Hierarchy

- *Item*
  - CTabItem

Syntax
```
'Declaration
Public Class CTabItem _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Selected	Whether the item is selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the visible tab.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects a tab.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ExpandBar Class

Description
The class for widgets that expand and close by clicking on a header.

Inheritance Hierarchy

- *Control*
  - ExpandBar

Syntax
```
'Declaration
Public Class ExpandBar _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
ExpandedItems	A list of the items that are expanded.
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)

Name	Description
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the expandbar.
Items	A list of items in the expandbar.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Collapse	Collapses the ExpandBar.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Expand	Expands an ExpandBar.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ExpandItem Class

Description
The class that contains the expandable items in an ExpandBar.

Inheritance Hierarchy

- *Item*
  - ExpandItem

Syntax

```
'Declaration
Public Class ExpandItem _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Content	The content of the item.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Expanded	Whether the item is expanded.
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Collapse	Collapses the expandable items.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Expand	Expands the expandable items.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)

Name	Description
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

HorizontalSash Class

Description
The class for the horizontal border around a window pane that can be dragged to adjust the window size.

Inheritance Hierarchy

- *Sash*
  - `HorizontalSash`

Syntax
```
'Declaration
Public Class HorizontalSash _
Inherits Sash
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)

Name	Description
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window pane that the sash borders. (Inherited from *Sash*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Sash Class

Description
The class for the border around a window pane that can be dragged to adjust the window size.

Inheritance Hierarchy

- *Control*
  - Sash
    - *HorizontalSash*
    - *VerticalSash*

Syntax

```
'Declaration
Public Class Sash _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window pane that the sash borders.
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SashForm Class

Description
The class for the border around a form that can be dragged to adjust the size of the form.

Inheritance Hierarchy
- *Control*
  - SashForm

Syntax
```
'Declaration
Public Class SashForm _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
Horizontal	Whether the sash form is horizontal.
MaximizedControl	The sash window that is maximized in the form.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Vertical	Whether the sash form is vertical.
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
Weights	The relative sizes of the children of the sash form window.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ScrollableControl Class

Description
The implementation class for all controls which are only classified as controls but also contain a scrollbar.

Inheritance Hierarchy
- *Control*
  - ScrollableControl

Syntax
```
'Declaration
Public Class ScrollableControl _
Inherits Control
```

Properties
Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ScrolledComposite Class

Description
The class for scrollbars and scrolls the content when the user uses the scrollbars.

Inheritance Hierarchy
- *Control*
  - ScrolledComposite

Syntax
```
'Declaration
Public Class ScrolledComposite _
Inherits Control
```

Properties

Name	Description
AlwaysShowScrollBars	Whether the scrollbar shows.
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Content	The content of the areas that can be dynamically positioned.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Origin	The point in the content that currently appears in the top left corner of the scrolled composite.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Shell Class

Description
The class for Java SWT windows.

Inheritance Hierarchy

- *Window*
  - Shell

Syntax

```
'Declaration
Public Class Shell _
Inherits Window
```

Properties

Name	Description
Application	The name of the Application that this Window belongs to. (Inherited from *Window*)
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Name	Description
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)

Name	Description
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use

Name	Description
	WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Spinner Class

Description
The class to enter and modify numeric values for up/down controls.

Inheritance Hierarchy

- *Control*
  - Spinner

Syntax

```
'Declaration
Public Class Spinner _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
IncrementCount	The amount to increase the position value of the spinner control after pressing it.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	The internal page size of the spinner control.
Position	The position for the up/down control.
Range	The range of the spinner control.

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Decrement	Decreases the value when the spinner controls are pressed.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Increment	Increases the value when the spinner controls are pressed.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PageDecrement	Decreases the value when the page spinner controls are pressed.
PageIncrement	Increases the value when the page spinner controls are pressed.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetValue	Sets the value for the spinner control.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

StyledText Class

Description
The class for an editable user interface object that displays lines of text.

Inheritance Hierarchy

- *TextField*
  - StyledText

Syntax
```
'Declaration
Public Class StyledText _
Inherits TextField
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Editable	Whether the text is editable.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
IsPassword	Whether the control is a password text field. (Inherited from *TextField*)
MultiLine	Whether the control is multiline. (Inherited from *TextField*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	The position of the caret within the text field. (Inherited from *TextField*)
SelectedRange	The selected range within the text field. (Inherited from *TextField*)
SelectedText	The selected text within the text field. An empty string if no text is selected. (Inherited from *TextField*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearText	Removes all text from the text field. (Inherited from *TextField*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetMultiText	Returns the specified lines of text in the multi-line text field. (Inherited from *TextField*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetMultiText	Substitutes all or part of the lines in the multi-line text field. (Inherited from *TextField*)
SetPosition	Sets the insertion point in the text field. (Inherited from *TextField*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the specified range of the single-line or multi-line text field. (Inherited from *TextField*)
SetText	Substitutes new text for all or part of the text in the text field. (Inherited from *TextField*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SWTBrowser Class

Description
The class for the browser widget.

Inheritance Hierarchy

- *Control*
  - SWTBrowser

Syntax

```
'Declaration
Public Class SWTBrowser _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown.

Name	Description
	The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SWTDateTime Class

Description
The class for SWT date and time elements in a calendar, date, or time control.

Inheritance Hierarchy
- *Control*
  - SWTDateTime

Syntax
```
'Declaration
Public Class SWTDateTime _
Inherits Control
```

Properties
Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
CurrentDate	The current date.
Day	The day.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
Hours	The hour.
Minutes	The minute.
Month	The month.

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Seconds	The second.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
Year	The year.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Select	Selects a date and time.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SWTTabControl Class

Description
The class for SWT TabFolder.

Inheritance Hierarchy

- *TabControl*
  - SWTTabControl

Syntax
```
'Declaration
Public Class SWTTabControl _
Inherits TabControl
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the tab control. (Inherited from *TabControl*)
Items	A list of items in the tab control. (Inherited from *TabControl*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The index of the selected item. (Inherited from *TabControl*)
SelectedItem	The selected item. (Inherited from *TabControl*)
SelectedTabItem	The name of the selected item.
TabItems	A list of TabItems in the tab control.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Sets the current page to the specified page. (Inherited from *TabControl*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SWTTabItem Class

Description
The class for SWT TabItem.

Inheritance Hierarchy

- *Item*
  - SWTTabItem

Syntax

```
'Declaration
Public Class SWTTabItem _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Selected	Whether the item is selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Select	Selects a tab.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SWTTable Class

Description
The class for SWT tables.

Inheritance Hierarchy

- *Table*
  - SWTTable

Syntax

```
'Declaration
Public Class SWTTable _
Inherits Table
```

Properties

Name	Description
AllowsCheck	Whether the control can display a checkmark. (Inherited from *Table*)
AllowsMultiSelect	Whether the table supports selecting multiple items. (Inherited from *Table*)
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
ColumnCount	The number of columns in the table. (Inherited from *Table*)
ColumnItems	A list of all items in the column. (Inherited from *Table*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
RowCount	The number of rows in the table. (Inherited from *Table*)
RowItems	A list of all table rows in the table. (Inherited from *Table*)
SelectedIndices	The indices of the selected item(s). (Inherited from *Table*)
SelectedItems	The selected item(s). (Inherited from *Table*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box in the defined row of a table. (Inherited from *Table*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Double-clicks an item. (Inherited from *Table*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
ExtendSelectRow	Selects a range of rows. (Inherited from *Table*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
FocusRow	Focuses on a row in the table. (Inherited from *Table*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MultiSelectRow	Adds a row from the table to the set of selected rows. (Inherited from *Table*)
MultiUnselectRow	Removes a row from the set of selected rows. (Inherited from *Table*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SelectRow	Selects an row in the table. (Inherited from *Table*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box in the defined row of a table. (Inherited from *Table*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SWTTableColumn Class

Description
The class for columns in an SWT table.

Inheritance Hierarchy

- *TableColumn*
  - SWTTableColumn

Syntax
```
'Declaration
Public Class SWTTableColumn _
Inherits TableColumn
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
Width	The width of the column. (Inherited from *TableColumn*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Resize	Resizes a column. (Inherited from *TableColumn*)
Select	Selects a column. (Inherited from *TableColumn*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout

Name	Description
	parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SWTTableRow Class

Description
The class for rows in an SWT table.

Inheritance Hierarchy

- *TableRow*
  - SWTTableRow

Syntax
```
'Declaration
Public Class SWTTableRow _
Inherits TableRow
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Checked	Whether the check box in the row is checked. (Inherited from *TableRow*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)

Name	Description
ItemCount	The number of items in the row. (Inherited from *TableRow*)
Items	A list of items in the row. (Inherited from *TableRow*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SWTTree Class

Description
The class for SWT tree widgets.

Inheritance Hierarchy

- *Tree*

- SWTTree

Syntax

```
'Declaration
Public Class SWTTree _
Inherits Tree
```

Properties

Name	Description
AllowsCheck	Whether the control can display a checkmark. (Inherited from *Tree*)
AllowsMultiSelect	Whether the control supports selecting multiple items. (Inherited from *Tree*)
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
ColumnCount	The number of columns in the tree.
Columns	A list of columns in the tree.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the tree (including all children). (Inherited from *Tree*)
ItemPaths	A list of items in the tree (including all children). (Inherited from *Tree*)
Items	A list of items in the tree (including all children). (Inherited from *Tree*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The index of the first selected item. (Inherited from *Tree*)
SelectedIndices	The indices of the selected item(s). (Inherited from *Tree*)
SelectedItem	The name of the first selected item. (Inherited from *Tree*)
SelectedItems	The names of the selected item(s). (Inherited from *Tree*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
VisibleItemPaths	A list of visible items in the tree (including all children). (Inherited from *Tree*)
VisibleItems	A list of visible items in the tree (including all children). (Inherited from *Tree*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box. (Inherited from *Tree*)
Click	Clicks on the object. (Inherited from *IClickable*)
Collapse	Collapses an item in a treeview control. (Inherited from *Tree*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Double-clicks an item. (Inherited from *Tree*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Expand	Expands an object in a treeview control. (Inherited from *Tree*)
ExtendSelect	Selects a range of items by extending the selection.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetContentsOfColumn	Retrieves the contents of the specified column as a list. To retrieve the contents of any column greater than column one, you must specify the parameters rawMode and column.
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetItemPaths	Returns all items of the tree as a list of itempaths (Inherited from *Tree*)

Name	Description
GetItemRect	Returns the size and position of an item relative to the treeview control. (Inherited from *Tree*)
GetItemsOfColumn	Retrieve items of the specified column from Tree
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetSelectedTextOfColumn	Retrieves selected text from the specified column
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsExpandable	Checks if the given item can be expanded. (Inherited from *Tree*)
IsExpanded	Checks if the given item is expanded. (Inherited from *Tree*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MultiSelect	Adds an item from the tree to the set of selected items.
MultiUnselect	Removes an item in the tree from the set of selected items.
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item from the tree. (Inherited from *Tree*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box. (Inherited from *Tree*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a

Name	Description
	WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SWTTreeColumn Class

Description
The class for columns in a Tree widget.

Inheritance Hierarchy

- *Item*
  - SWTTreeColumn

Syntax

```
'Declaration
Public Class SWTTreeColumn _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
Width	The width of the column.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Resize	Resizes a column of the tree.
Select	Selects a tree column.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

VerticalSash Class

Description
The class for the vertical border around a window pane that can be dragged to adjust the window size.

Inheritance Hierarchy

- *Sash*
  - `VerticalSash`

Syntax
```
'Declaration
Public Class VerticalSash _
Inherits Sash
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)

Name	Description
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window pane that the sash borders. (Inherited from *Sash*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ViewForm Class

Description
The class to lay out a view's label/menu/toolbar local bar in the workbench.

Inheritance Hierarchy

- *Control*
  - ViewForm

Syntax
```
'Declaration
Public Class ViewForm _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Content	The content of the areas that can be dynamically positioned.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
TopCenter	The window located in the top center of the pane.
TopLeft	The window located in the top left of the pane.
TopRight	The window located in the top right of the pane.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Keyword-Driven Testing Class Reference

Lists the available classes for keyword-driven testing.

ArgumentAttribute Class

Description

Used to explicitly specify the name of an argument of a method that is marked as a keyword for keyword-driven testing.

Syntax

```
<AttributeUsageAttribute(AttributeTargets.Parameter>
Public Class ArgumentAttribute Inherits Attribute
```

Properties

Name	Description
Name Property (ArgumentAttribute)	Gets the name of the argument.

Example: Creating a keyword with two arguments

Use the following code sample to create the keyword *Login* with the arguments `username` and `password`:

```
<Keyword()>
Public Sub Login(<Argument("Name of the user")> username As
String, <Argument("Password of the user")> password As String)
  ' keyword implementation
End Sub
```

Example: Using arguments without a description

Use one of the following code samples to create the keyword *Login* with the arguments `username` and `password`, without using a description for the arguments:

```
<Keyword()>
Public Sub Login(username As String, password As String)
  ' keyword implementation
End Sub
```

```
<Keyword()>
Public Sub Login(<Argument> username As String, <Argument>
password As String)
  ' keyword implementation
End Sub
```

KeywordAttribute Class

Description

Specifies that a method is a keyword. If the name of the keyword is not specified the name of the annotated method is used as the keyword name.

Syntax

```
'Declaration
<AttributeUsageAttribute(AttributeTargets.Method>
Public Class KeywordAttribute Inherits Attribute
```

Properties

Name	Description
Name Property (KeywordAttribute)	Gets the name of the keyword. The default value is an empty string.
Description Property (KeywordAttribute)	Gets the description of the keyword. The default value is an empty string.

Example: Creating a keyword with two arguments

Use the following code sample to create the keyword *Login* with the arguments `username` and `password`:

```
<Keyword()>
Public Sub Login(username As String, password As String)
  ' keyword implementation
End Sub
```

Example: Specifying a keyword name

To specify the keyword name when creating a new keyword, use the following code sample:

```
<Keyword("Login user")>
Public Sub Login(username As String, password As String)
  ' keyword implementation
End Sub
```

KeywordGroupAttribute Class

Description

Specifies that a class is a keyword group. Methods within this class that are marked as keywords are keywords. The qualified name of these methods consists of the name of the keyword group and the name of the keyword. If the keyword group has no name, the simple name of the declaring class is used instead.

Syntax

```
'Declaration
<AttributeUsageAttribute(AttributeTargets.Class>
Public Class KeywordGroupAttribute Inherits Attribute
```

Properties

Name	Description
Name Property (KeywordGroupAttribute)	Gets the name of the keyword group. The default value is an empty string.

Example: Creating a keyword group

Use the following code sample to create the keyword group *LoginPage*:

```
<KeywordGroup()>
Public Class LoginPage
  ' keyword methods
End Class
```

Example: Specifying the name of the keyword group

Use one of the following code samples to create the keyword *Login* with the arguments `username` and `password`, without using a description for the arguments:

```
<KeywordGroup("Login Page")>
Public Class LoginPage
  ' keyword methods
End Class
```

Mobile Class Reference

Lists the available classes for testing Mobile controls.

IMobileClickable Interface

Description
Interface for objects that use clicks.

Syntax

```
'Declaration
Public Class IMobileClickable
```

Methods

Name	Description
Click	Clicks on the device at the specified coordinates.
DoubleClick	Double-clicks on the device at the specified coordinates.
LongClick	Long-clicks on the device at the specified coordinates.

IMobileGestures Interface

Description
Interface for objects that use long swipes.

Syntax

```
'Declaration
Public Class IMobileGestures
```

Methods

Name	Description
Drag	Performs a drag operation between the two specified points.
MultiTouch	Performs a multi-touch operation with the given pointers. For every given pointer a swipe operation is executed at the same time.
PinchIn	Performs a two-pointer gesture, where each pointer moves toward the other, from the edges to the center of this object. For example to zoom out on an image.
PinchOut	Performs a two-pointer gesture, where each pointer moves opposite across the other, from the center out towards the edges of the this object. For example to zoom in on an image.
Swipe	Performs a swipe between the two specified points.
SwipeDown	Performs a down-swipe.
SwipeLeft	Performs a left-swipe.

Name	Description
SwipeRight	Performs a right-swipe.
SwipeUp	Performs a up-swipe.
SwipeWithSegments	Performs a swipe between the specified points.
TwoPointerMultiTouch	Performs a multi-touch operation with two pointers. It executes two swipe operation at the same time.

IMobileKeyable Interface

Description
Interface for objects that use keystrokes.

Syntax
```
'Declaration
Public Class IMobileKeyable
```

Methods

Name	Description
TypeKeys	Sends a set of keystrokes to the object.

MobileButton Class

Description
Represents a mobile button. Mobile buttons can be clicked.

Inheritance Hierarchy

- *MobileObject*
  - MobileButton

Syntax
```
'Declaration
Public Class MobileButton _
Inherits MobileObject
```

Properties

Name	Description
IsEnabled	Whether the mobile object is enabled. (Inherited from *MobileObject*)
IsFocused	Whether the mobile object has focus. This property is not supported on iOS. (Inherited from *MobileObject*)
IsSelected	Whether the mobile object is selected or checked. (Inherited from *MobileObject*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
DoubleClick	Double-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
Drag	Performs a drag operation between the two specified points. (Inherited from *IMobileGestures*)
DragTo	Drags this object to the specified point relative to the mobile device. (Inherited from *MobileObject*)
DragToObject	Drags this object to the specified destination object. (Inherited from *MobileObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
LongClick	Long-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
MultiTouch	Performs a multi-touch operation with the given pointers. For every given pointer a swipe operation is executed at the same time. (Inherited from *IMobileGestures*)
PinchIn	Performs a two-pointer gesture, where each pointer moves toward the other, from the edges to the center of this object. For example to zoom out on an image. (Inherited from *IMobileGestures*)
PinchOut	Performs a two-pointer gesture, where each pointer moves opposite across the other, from the center out towards the edges of the this object. For example to zoom in on an image. (Inherited from *IMobileGestures*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Swipe	Performs a swipe between the two specified points. (Inherited from *IMobileGestures*)
SwipeDown	Performs a down-swipe. (Inherited from *IMobileGestures*)
SwipeLeft	Performs a left-swipe. (Inherited from *IMobileGestures*)
SwipeRight	Performs a right-swipe. (Inherited from *IMobileGestures*)
SwipeUp	Performs a up-swipe. (Inherited from *IMobileGestures*)

Name	Description
SwipeWithSegments	Performs a swipe between the specified points. (Inherited from *IMobileGestures*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TwoPointerMultiTouch	Performs a multi-touch operation with two pointers. It executes two swipe operation at the same time. (Inherited from *IMobileGestures*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IMobileKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

MobileDevice Class

Description
The mobile device.

Inheritance Hierarchy

- *TestObject*
  - MobileDevice

Syntax
```
'Declaration
Public Class MobileDevice _
Inherits TestObject _
Implements IMobileClickable, IMobileGestures, IMobileKeyable
```

Properties

Name	Description
DeviceId	The id of the device for the current connection.
DisplayHeight	The height of the display in pixels.
DisplayOrientation	The current orientation of the device.
DisplayWidth	The width of the display in pixels.
IsEmulator	Whether the connected device is an emulator.
Model	The device name of the connected device.
OperatingSystem	the operating system of the mobile device, Android or iOS
OsVersion	The version of the operating system of the connected device.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)

Name	Description
CloseApp	Closes the app and releases the device, to make it available to other users.
DoubleClick	Double-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
Drag	Performs a drag operation between the two specified points. (Inherited from *IMobileGestures*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateConnectionString	Generates a connection string for this mobile device.
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
LongClick	Long-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
MultiTouch	Performs a multi-touch operation with the given pointers. For every given pointer a swipe operation is executed at the same time. (Inherited from *IMobileGestures*)
PinchIn	Performs a two-pointer gesture, where each pointer moves toward the other, from the edges to the center of this object. For example to zoom out on an image. (Inherited from *IMobileGestures*)
PinchOut	Performs a two-pointer gesture, where each pointer moves opposite across the other, from the center out towards the edges of the this object. For example to zoom in on an image. (Inherited from *IMobileGestures*)
PressBack	Presses the back button. This method is not supported on iOS.
PressDelete	Presses the delete button.
PressEnter	Presses the enter button.
PressHome	Presses the home button.
PressKeyCode	Presses a certain Android-specific key code. This method is not supported on iOS.
PressRecentApps	Presses the recent apps button.
Rotate	Simulates rotating the device screen.
SetLocation	Sets the geographic location of the device, this method only works on Android devices (physical or emulated) and iOS simulators.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Sleep	Presses the power button if the screen is on. This method is not supported on iOS.
Swipe	Performs a swipe between the two specified points. (Inherited from *IMobileGestures*)
SwipeDown	Performs a down-swipe. (Inherited from *IMobileGestures*)
SwipeLeft	Performs a left-swipe. (Inherited from *IMobileGestures*)
SwipeRight	Performs a right-swipe. (Inherited from *IMobileGestures*)
SwipeUp	Performs a up-swipe. (Inherited from *IMobileGestures*)
SwipeWithSegments	Performs a swipe between the specified points. (Inherited from *IMobileGestures*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TwoPointerMultiTouch	Performs a multi-touch operation with two pointers. It executes two swipe operation at the same time. (Inherited from *IMobileGestures*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IMobileKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WakeUp	Presses the power button if the screen is off. This method is not supported on iOS.

MobileObject Class

Description
The base class for all mobile objects.

Inheritance Hierarchy

- *TestObject*
  - MobileObject
    - *MobileButton*
    - *MobileTextField*
    - *MobileWindow*

Syntax
```
'Declaration
Public Class MobileObject _
Inherits TestObject _
Implements IMobileClickable, IMobileGestures, IMobileKeyable
```

Properties

Name	Description
IsEnabled	Whether the mobile object is enabled.
IsFocused	Whether the mobile object has focus. This property is not supported on iOS.
IsSelected	Whether the mobile object is selected or checked.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
DoubleClick	Double-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)

Name	Description
Drag	Performs a drag operation between the two specified points. (Inherited from *IMobileGestures*)
DragTo	Drags this object to the specified point relative to the mobile device.
DragToObject	Drags this object to the specified destination object.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
LongClick	Long-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
MultiTouch	Performs a multi-touch operation with the given pointers. For every given pointer a swipe operation is executed at the same time. (Inherited from *IMobileGestures*)
PinchIn	Performs a two-pointer gesture, where each pointer moves toward the other, from the edges to the center of this object. For example to zoom out on an image. (Inherited from *IMobileGestures*)
PinchOut	Performs a two-pointer gesture, where each pointer moves opposite across the other, from the center out towards the edges of the this object. For example to zoom in on an image. (Inherited from *IMobileGestures*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Swipe	Performs a swipe between the two specified points. (Inherited from *IMobileGestures*)
SwipeDown	Performs a down-swipe. (Inherited from *IMobileGestures*)
SwipeLeft	Performs a left-swipe. (Inherited from *IMobileGestures*)
SwipeRight	Performs a right-swipe. (Inherited from *IMobileGestures*)
SwipeUp	Performs a up-swipe. (Inherited from *IMobileGestures*)
SwipeWithSegments	Performs a swipe between the specified points. (Inherited from *IMobileGestures*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TwoPointerMultiTouch	Performs a multi-touch operation with two pointers. It executes two swipe operation at the same time. (Inherited from *IMobileGestures*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IMobileKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

MobileTextField Class

Description
Represents a text field.

Inheritance Hierarchy

- *MobileObject*
  - MobileTextField

Syntax
```
'Declaration
Public Class MobileTextField _
Inherits MobileObject
```

Properties

Name	Description
IsEnabled	Whether the mobile object is enabled. (Inherited from *MobileObject*)

Name	Description
IsFocused	Whether the mobile object has focus. This property is not supported on iOS. (Inherited from *MobileObject*)
IsSelected	Whether the mobile object is selected or checked. (Inherited from *MobileObject*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearText	Clears all the content from the text field.
Click	Clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
DoubleClick	Double-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
Drag	Performs a drag operation between the two specified points. (Inherited from *IMobileGestures*)
DragTo	Drags this object to the specified point relative to the mobile device. (Inherited from *MobileObject*)
DragToObject	Drags this object to the specified destination object. (Inherited from *MobileObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
LongClick	Long-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
MultiTouch	Performs a multi-touch operation with the given pointers. For every given pointer a swipe operation is executed at the same time. (Inherited from *IMobileGestures*)
PinchIn	Performs a two-pointer gesture, where each pointer moves toward the other, from the edges to the center of this object. For example to zoom out on an image. (Inherited from *IMobileGestures*)
PinchOut	Performs a two-pointer gesture, where each pointer moves opposite across the other, from the center out towards the edges of the this object. For example to zoom in on an image. (Inherited from *IMobileGestures*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
SetText	Substitutes new text for the text in the text field.
Swipe	Performs a swipe between the two specified points. (Inherited from *IMobileGestures*)
SwipeDown	Performs a down-swipe. (Inherited from *IMobileGestures*)
SwipeLeft	Performs a left-swipe. (Inherited from *IMobileGestures*)
SwipeRight	Performs a right-swipe. (Inherited from *IMobileGestures*)
SwipeUp	Performs a up-swipe. (Inherited from *IMobileGestures*)
SwipeWithSegments	Performs a swipe between the specified points. (Inherited from *IMobileGestures*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TwoPointerMultiTouch	Performs a multi-touch operation with two pointers. It executes two swipe operation at the same time. (Inherited from *IMobileGestures*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IMobileKeyable*)
TypePasswordKeys	Types an encrypted password into a text field.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

MobileWindow Class

Description
The mobile window.

Inheritance Hierarchy

- *MobileObject*
  - MobileWindow

Syntax

```
'Declaration
Public Class MobileWindow _
Inherits MobileObject
```

Properties

Name	Description
IsEnabled	Whether the mobile object is enabled. (Inherited from *MobileObject*)
IsFocused	Whether the mobile object has focus. This property is not supported on iOS. (Inherited from *MobileObject*)
IsSelected	Whether the mobile object is selected or checked. (Inherited from *MobileObject*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
DoubleClick	Double-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
Drag	Performs a drag operation between the two specified points. (Inherited from *IMobileGestures*)
DragTo	Drags this object to the specified point relative to the mobile device. (Inherited from *MobileObject*)
DragToObject	Drags this object to the specified destination object. (Inherited from *MobileObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
LongClick	Long-clicks on the device at the specified coordinates. (Inherited from *IMobileClickable*)
MultiTouch	Performs a multi-touch operation with the given pointers. For every given pointer a swipe operation is executed at the same time. (Inherited from *IMobileGestures*)
PinchIn	Performs a two-pointer gesture, where each pointer moves toward the other, from the edges to the center of this object. For example to zoom out on an image. (Inherited from *IMobileGestures*)
PinchOut	Performs a two-pointer gesture, where each pointer moves opposite across the other, from the center out towards the edges of the this object. For example to zoom in on an image. (Inherited from *IMobileGestures*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Swipe	Performs a swipe between the two specified points. (Inherited from *IMobileGestures*)
SwipeDown	Performs a down-swipe. (Inherited from *IMobileGestures*)
SwipeLeft	Performs a left-swipe. (Inherited from *IMobileGestures*)
SwipeRight	Performs a right-swipe. (Inherited from *IMobileGestures*)
SwipeUp	Performs a up-swipe. (Inherited from *IMobileGestures*)
SwipeWithSegments	Performs a swipe between the specified points. (Inherited from *IMobileGestures*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TwoPointerMultiTouch	Performs a multi-touch operation with two pointers. It executes two swipe operation at the same time. (Inherited from *IMobileGestures*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IMobileKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Rumba Class Reference

Lists the available classes for testing Rumba controls.

Rumba Keys

Description
Contains the string constants for the Rumba key mnemonics which can be used in the `SendKeys` and `SynchSendKeys` methods of the `RumbaObject` class.

Includes the following string constants:

String Constant	Key Mnemonic
At	@@
Alt	@A
AltCursor	@e@1*
Attention	@A@Q
Backspace	@B
Backtab	@B
Clear	@C
Cmd	@A@Y
CursorDown	@V
CursorLeft	@L
CursorRight	@Z
CursorSelect	@e@1*
CursorUp	@U
Delete	@D
Dup	@S@x
End	@q
Enter	@E
EraseEof	@F
EraseEol	@e@d
EraseInput	@A@F
FieldExit	@A@E
FieldMark	@S@y
FieldMinus	@A@-
FieldPlus	@A@+
Help	@H
Hexadecimal	@A@X
Home	@0
Insert	@I
InsertToggle	@A@I
LocalHelp	@e@l
LocalPrint	@P
LocalHelpCursor	@e@m*
LeftTab	@B
NewLine	@e@n*
PageUp	@u
PageDown	@v

String Constant	Key Mnemonic
Print	@A@t
PrintScreen	@A@T
RecordBackspace	@A@<
Reset	@R
RightTab	@T
Shift	@S
SysRequest	@A@H
Tab	@T
Test	@A@C
Usm	@e@v
AplCutOnOf	@e@2
PA1	@x
PA2	@y
PA3	@z
PA4	@+
PA5	@%
PA6	@&
PA7	@'
PA8	@(
PA9	@)
PA10	@*
PF1	@1
PF2	@2
PF3	@3
PF4	@4
PF5	@5
PF6	@6
PF7	@7
PF8	@8
PF9	@9
PF10	@a
PF11	@b
PF12	@c
PF13	@d
PF14	@e
PF15	@f

String Constant	Key Mnemonic
PF16	@g
PF17	@h
PF18	@i
PF19	@j
PF20	@k
PF21	@l
PF22	@m
PF23	@n
PF24	@o
Cents	@n@1
VerticalBar	@d@2
Corner	@d@3
Plus	@d@4
Minus	@d@5
Quote	@d@7
HorizontalBar	@d@8
OpenBracket	@d@9
CloseBracket	@d@10
Substitute	@d@0
Synchronous	@d@a

RumbaCharacterAttribute Data Type

Represents the character attributes of an on-screen character.

Can include the following character attributes:

Attribute	Description
backgroundColor	The background color of the character.
foregroundColor	The foreground color of the character.
column	The column of the character on the screen.
line	The line of the character on the screen.
autoEnter	Defines if the character is part of an auto-enter text.
autoSkip	Defines if the character is part of an auto-skip text.
blinking	Defines if the character is blinking.
bold	Defines if the character is bold.
columnSeparator	Defines if the character is a column separator.
detectable	Defines if the character is part of a detectable text.
hidden	Defines if the character is part of a hidden text.

Attribute	Description
protect	Defines if the character is part of a protected text.
reverseVideo	Defines if the character is displayed in reverse video.
underlined	Defines if the character is underlined.

For additional information, see *IRDEAttribute Properties* in the Rumba Developer Edition (RDE) documentation.

RumbaField Class

Description
Represents a field on the green screen.

Inheritance Hierarchy

- *RumbaObject*
  - RumbaField
    - *RumbaLabel*
    - *RumbaTextField*

Syntax

```
'Declaration
Public Class RumbaField _
Inherits RumbaObject
```

Properties

Name	Description
AutoEnter	A value that indicates true if the field is an AutoEnter field.
AutoSkip	A value that is true if the field is an AutoSkip field.
Detectable	A value is indicates true if the field is detectable.
Dup	A value that indicates true if the field is a dup field.
EndColumn	A value that indicates the ending column of the field.
EndLine	A value that indicates the ending line of the field.
EndPosition	The 1-based end position of the field, when the screen is seen as one sequence of characters.
Enptui	A value that indicates true if the field is an ENPTUI field.
FieldExitRequired	A value that indicates true if the field is required before exiting the screen.
Hidden	A value that indicates if the field is hidden.
Intense	A value that indicates true if the field is intense.
Length	The length of the field.
ModifiedDataTag	A value that indicates true if the field has the ModifiedDataTag property set.

Name	Description
Protected	A value that indicates true if the field is protected (i.e. a RumbaLabel).
StartColumn	A value that indicates the column in which the field begins.
StartLine	A value that indicates the line in which the field begins.
StartPosition	The 1-based start position of the field, when the screen is seen as one sequence of characters.
Text	The text of the control. (Inherited from *TestObject*)
Type	A value that represents the input allowed for this field.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCharacterAttributes	Gets the character attributes for this field's characters.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SendKeys	Sends keys. Supports RDE mnemonics. (Inherited from *RumbaObject*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
SynchSendKeys	Synchronously sends keys. Supports RDE mnemonics. (Inherited from *RumbaObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

RumbaLabel Class

Description
A field in the screen whose text can be only be retrieved but not set.

Inheritance Hierarchy

- *RumbaField*
  - RumbaLabel

Syntax
```
'Declaration
Public Class RumbaLabel _
Inherits RumbaField
```

Properties

Name	Description
AutoEnter	A value that indicates true if the field is an AutoEnter field. (Inherited from *RumbaField*)
AutoSkip	A value that is true if the field is an AutoSkip field. (Inherited from *RumbaField*)
Detectable	A value is indicates true if the field is detectable. (Inherited from *RumbaField*)
Dup	A value that indicates true if the field is a dup field. (Inherited from *RumbaField*)
EndColumn	A value that indicates the ending column of the field. (Inherited from *RumbaField*)
EndLine	A value that indicates the ending line of the field. (Inherited from *RumbaField*)
EndPosition	The 1-based end position of the field, when the screen is seen as one sequence of characters. (Inherited from *RumbaField*)
Enptui	A value that indicates true if the field is an ENPTUI field. (Inherited from *RumbaField*)
FieldExitRequired	A value that indicates true if the field is required before exiting the screen. (Inherited from *RumbaField*)
Hidden	A value that indicates if the field is hidden. (Inherited from *RumbaField*)
Intense	A value that indicates true if the field is intense. (Inherited from *RumbaField*)
Length	The length of the field. (Inherited from *RumbaField*)
ModifiedDataTag	A value that indicates true if the field has the ModifiedDataTag property set. (Inherited from *RumbaField*)
Protected	A value that indicates true if the field is protected (i.e. a RumbaLabel). (Inherited from *RumbaField*)

Name	Description
StartColumn	A value that indicates the column in which the field begins. (Inherited from *RumbaField*)
StartLine	A value that indicates the line in which the field begins. (Inherited from *RumbaField*)
StartPosition	The 1-based start position of the field, when the screen is seen as one sequence of characters. (Inherited from *RumbaField*)
Text	The text of the control. (Inherited from *TestObject*)
Type	A value that represents the input allowed for this field. (Inherited from *RumbaField*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCharacterAttributes	Gets the character attributes for this field's characters. (Inherited from *RumbaField*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SendKeys	Sends keys. Supports RDE mnemonics. (Inherited from *RumbaObject*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SynchSendKeys	Synchronously sends keys. Supports RDE mnemonics. (Inherited from *RumbaObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

RumbaObject Class

Description
RumbaObject.

Inheritance Hierarchy

- *TestObject*
  - RumbaObject
    - *RumbaField*
    - *RumbaScreen*

Syntax

```
'Declaration
Public Class RumbaObject _
Inherits TestObject _
Implements IClickable, IFocusable, IKeyable
```

Properties

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SendKeys	Sends keys. Supports RDE mnemonics.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SynchSendKeys	Synchronously sends keys. Supports RDE mnemonics.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

RumbaScreen Class

Description
Represents the whole green screen.

Inheritance Hierarchy

- *RumbaObject*
  - RumbaScreen

Syntax

```
'Declaration
Public Class RumbaScreen _
Inherits RumbaObject
```

Properties

Name	Description
BackgroundColor	the background color of the display control.
CharacterSetId	a value that identifies the character set translation table used by the Display control.
ColumnCount	a value that indicates the number of columns on the current screen.
CursorColumn	the column position of the cursor on the screen.
CursorLine	the line position of the cursor on the screen.
CursorPosition	The current cursor position on the screen. Values are starting with 1 for the first line or column.
HostConnected	a value that indicates if the host is connected.
HostDeviceName	the host device name.
HostIpAddress	the host IP address.
KeyboardLocked	a value that indicates whether keyboard input is inhibited.
LineCount	a value that indicates the number of lines on the current screen.
ScreenId	the screen id.
ScreenSize	the size of the screen in text lines and columns.
SessionName	the session name.

Name	Description
SupportsStructuredFields	a value that indicates if the host supports structured field decoding.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Clears the screen by sending the "clear" key. Equivalent to synchSendKeys(RumbaKey.CLEAR).
Click	Clicks on the object. (Inherited from *IClickable*)
Connect	Connects to the hosts in the current configuration.
Disconnect	Disconnects the current connection.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCharacterAttributes	Gets the character attributes for length-number of characters starting at the given position.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetContent	Retrieves a string containing the screen without line breaks starting at the specified text position.

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetLines	Retrieves a clipping of the screen and returns the selected lines in a list of strings.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
Reset	Resets the screen by sending the "reset" key. Equivalent to synchSendKeys(RumbaKey.RESET).
SendKeys	Sends keys. Supports RDE mnemonics. (Inherited from *RumbaObject*)
SetCursorPosition	Moves the cursor to the given position of the screen.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Submit	Submits the entered data to the server by sending the "submit" key. Equivalent to synchSendKeys(RumbaKey.SUBMIT).
SynchSendKeys	Synchronously sends keys. Supports RDE mnemonics. (Inherited from *RumbaObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForHostConnected	Waits until the session is connected.
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a

Name	Description
	WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WriteText	Writes text on the screen.

RumbaTextField Class

Description
A field in the screen whose text can be both retrieved and set.

Inheritance Hierarchy

- *RumbaField*
  - RumbaTextField

Syntax
```
'Declaration
Public Class RumbaTextField _
Inherits RumbaField
```

Properties

Name	Description
AutoEnter	A value that indicates true if the field is an AutoEnter field. (Inherited from *RumbaField*)
AutoSkip	A value that is true if the field is an AutoSkip field. (Inherited from *RumbaField*)
Detectable	A value is indicates true if the field is detectable. (Inherited from *RumbaField*)
Dup	A value that indicates true if the field is a dup field. (Inherited from *RumbaField*)
EndColumn	A value that indicates the ending column of the field. (Inherited from *RumbaField*)
EndLine	A value that indicates the ending line of the field. (Inherited from *RumbaField*)
EndPosition	The 1-based end position of the field, when the screen is seen as one sequence of characters. (Inherited from *RumbaField*)
Enptui	A value that indicates true if the field is an ENPTUI field. (Inherited from *RumbaField*)

Name	Description
FieldExitRequired	A value that indicates true if the field is required before exiting the screen. (Inherited from *RumbaField*)
Hidden	A value that indicates if the field is hidden. (Inherited from *RumbaField*)
Intense	A value that indicates true if the field is intense. (Inherited from *RumbaField*)
Length	The length of the field. (Inherited from *RumbaField*)
ModifiedDataTag	A value that indicates true if the field has the ModifiedDataTag property set. (Inherited from *RumbaField*)
Protected	A value that indicates true if the field is protected (i.e. a RumbaLabel). (Inherited from *RumbaField*)
StartColumn	A value that indicates the column in which the field begins. (Inherited from *RumbaField*)
StartLine	A value that indicates the line in which the field begins. (Inherited from *RumbaField*)
StartPosition	The 1-based start position of the field, when the screen is seen as one sequence of characters. (Inherited from *RumbaField*)
Text	The text of the control. (Inherited from *TestObject*)
Type	A value that represents the input allowed for this field. (Inherited from *RumbaField*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCharacterAttributes	Gets the character attributes for this field's characters. (Inherited from *RumbaField*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SendKeys	Sends keys. Supports RDE mnemonics. (Inherited from *RumbaObject*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetText	Sets the text field's text.
SynchSendKeys	Synchronously sends keys. Supports RDE mnemonics. (Inherited from *RumbaObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SAP Class Reference

Lists the available classes for testing SAP controls.

ISapContextMenuable Interface

Description
Interface for any Component which supports ContextMenu selections.

Syntax

```
'Declaration
Public Class ISapContextMenuable
```

Methods

Name	Description
SelectContextMenuItem	Select an item from the controls context menu.
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item.
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus.

SapBarChart Class

Description
Class for SAP Bar Chart.

Inheritance Hierarchy

- *SapShell*
  - SapBarChart

Syntax

```
'Declaration
Public Class SapBarChart _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityDescription	the accessibility description of the shell.
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
ChartCount	the number of charts.
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)

Name	Description
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)
WindowHandle	the window handle of the control that is connected to the GuiShell.

Methods

Name	Description
BarCount	Returns the number of bars in the given chart.
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetBarContent	Returns the content of the bar.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetGridLineContent	Returns the content of the grid line.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GridCount	Returns the number of grids within the chart.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
LinkCount	Returns the number of links within the given chart.
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SendData	Send data to the server.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapBox Class

Description
Class for SAP SapBox.

Inheritance Hierarchy

- *SapComponent*
  - SapBox

Syntax

```
'Declaration
Public Class SapBox _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapButton Class

Description
SapButton class is the class for push buttons.

Inheritance Hierarchy

- *SapComponent*
  - SapButton

Syntax

```
'Declaration
Public Class SapButton _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CharHeight	the height of the element in character metric.
CharLeft	the left coordinate of the element in character metric.
CharTop	the top coordinate of the element in character metric.
CharWidth	the width of the element in character metric.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)

Name	Description
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Select	This emulates manually pressing a button. Pressing a button will always cause server communication to occur, rendering all references to elements below the window level invalid.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapCalendar Class

Description
SapCalendar class represents a SapCalendar.

Inheritance Hierarchy

- *SapShell*
  - SapCalendar

Syntax

```
'Declaration
Public Class SapCalendar _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
Day	The day of the month that is currently selected.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
EndSelection	The end of the selection interval.
FirstVisibleDate	the earliest date visible in the calendar control.
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
LastVisibleDate	the last date visible in the calendar control.
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Month	The month of the year that is currently selected.
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
SelectionInterval	The interval that is currently selected.
StartSelection	The start of the selection interval.
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Week	The week of the year that is currently selected.
Width	the current width of the component. (Inherited from *SapComponent*)
Year	The year that is currently selected.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ContextMenu	Opens a context menu.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SetFirstVisibleDate	Sets the earliest date visible in the calendar control.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetFocusDate	Sets the date to be focused.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionInterval	Selects the specified interval.
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapChart Class

Description
Class for SAP Chart.

Inheritance Hierarchy

- *SapShell*
  - SapChart

Syntax

```
'Declaration
Public Class SapChart _
```

```
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityDescription	the accessibility description of the shell.
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Width	the current width of the component. (Inherited from *SapComponent*)
WindowHandle	the window handle of the control that is connected to the GuiShell.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
ValueChange	Changes a value of the chart.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapCheckBox Class

Description
SapCheckBox class is the class for check box controls.

Inheritance Hierarchy

- *SapComponent*
  - SapCheckBox

Syntax
```
'Declaration
Public Class SapCheckBox _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)

Name	Description
CharHeight	the height of the element in character metric.
CharLeft	the left coordinate of the element in character metric.
CharTop	the top coordinate of the element in character metric.
CharWidth	the width of the element in character metric.
Checked	whether the checkbox is checked.
ColorIndex	the index of the list color of this element.
ColorIntensified	whether the Intensified flag is set in screen painter for this element.
ColorInverse	whether the inverse color style is set in screen painter for the element.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Flushing	whether the value change causes a round trip.
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsLeftLabel	whether the component has the 'assign left' flag.
IsListElement	whether the element is on an ABAP list, not a screen.
IsRightLabel	whether the component has the 'assign right' flag.
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
RowText	the text of the while line containing the current component.
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Uncheck	Un-checks the check box.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapColorSelector Class

Description
SapColorSelector class represents a SapColorSelector.

Inheritance Hierarchy

- *SapShell*
  - SapColorSelector

Syntax
```
'Declaration
Public Class SapColorSelector _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)

Name	Description
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeSelection	Selects the color at the given index position.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)

Name	Description
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapComboBox Class

Description
Class for SAP ComboBox.

Inheritance Hierarchy

- *SapComponent*
  - SapComboBox

Syntax
```
'Declaration
Public Class SapComboBox _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CharHeight	the height of the element in character metric.
CharLeft	the left coordinate of the element in character metric.
CharTop	the top coordinate of the element in character metric.
CharWidth	the width of the element in character metric.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Flushing	whether the value change causes a round trip.
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsLeftLabel	whether the component has the 'assign left' flag.
IsListBoxActive	whether the list box of the the combo box is currently open.

Name	Description
IsRightLabel	whether the component has the 'assign right' flag.
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Key	the key of the currently selected item.
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
Required	whether the component is a required value for the screen.
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCurrentEntryKey	the key of the current entry.
GetCurrentEntryValue	the value of the current entry.
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetEntryKeys	a list of all entry keys.
GetEntryValues	a list of all entry values.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SelectItem	Selects the specified item.
SelectKey	Selects the list item with the specified key.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapComponent Class

Description
SapComponent class is the base class for SAP.

Inheritance Hierarchy

- *TestObject*
  - SapComponent
    - *SapBox*
    - *SapButton*
    - *SapCheckBox*
    - *SapComboBox*
    - *SapContainer*
    - *SapContainerShell*
    - *SapCustomControl*
    - *SapDockShell*
    - *SapLabel*
    - *SapMenu*
    - *SapMenubar*
    - *SapOkCodeField*
    - *SapRadioButton*
    - *SapScrollContainer*
    - *SapShell*
    - *SapSimpleContainer*
    - *SapSplitterContainer*
    - *SapStatusbar*
    - *SapTab*
    - *SapTable*
    - *SapTabStrip*
    - *SapTextField*
    - *SapTitlebar*
    - *SapToolbar*
    - *SapUserArea*
    - *SapWindow*

Syntax

```
'Declaration
Public Class SapComponent _
Inherits TestObject _
Implements IFocusable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support.
AccessibilityTextOnRequest	an additional text for accessibility support.
AccessibilityTooltip	an additional tooltip text for accessibility support.
Changeable	whether an object is changeable.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type.
Height	the current height of the component.

Name	Description
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string.
IsSymbolFont	whether the component's text is visualized in the SAP symbol font.
Left	the left position of the element in screen coordinates.
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system.
Name	the name of the component.
ScreenLeft	the left position of the component in screen coordinates.
ScreenTop	the top position of the component in screen coordinates.
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text.
Top	the top coordinate of the element in screen coordinates.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapContainer Class

Description
Class for SAP Container.

Inheritance Hierarchy

- *SapComponent*
  - SapContainer

Syntax

```
'Declaration
Public Class SapContainer _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapContainerShell Class

Description
Class for SAP ConainerShell.

Inheritance Hierarchy

- *SapComponent*
  - SapContainerShell

Syntax

```
'Declaration
Public Class SapContainerShell _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)

Name	Description
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapContextMenu Class

Description

The class for SAP context menus.

For information on determining the items of a SAP context menu, see *Determining the item names and item texts of a SapContextMenu*.

Inheritance Hierarchy

- *SapMenu*
  - SapContextMenu

Syntax

```
'Declaration
Public Class SapContextMenu _
Inherits SapMenu
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)

Name	Description
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Select	Select the menu. (Inherited from *SapMenu*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapCustomControl Class

Description
Class for SAP CustomControl.

Inheritance Hierarchy

- *SapComponent*
  - SapCustomControl

Syntax

```
'Declaration
Public Class SapCustomControl _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)

Name	Description
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapDockShell Class

Description
Class for SAP DockShell.

Inheritance Hierarchy

- *SapComponent*
  - SapDockShell

Syntax

```
'Declaration
Public Class SapDockShell _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)

Name	Description
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapGridView Class

Description
Class for SAP GridView. Some of the methods that have Get as a prefix require the column as an iput parameter of type string. This parameter is different for each individual GridView element. To obtain the value of this parameter, record an action on the desired item and use the parameter that you have recorded.

Inheritance Hierarchy

- *SapShell*
  - SapGridView

Syntax

```
'Declaration
Public Class SapGridView _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
ColumnCount	the number of columns.
ColumnOrder	a list of strings which contains all the column identifiers in the order in which they are displayed.
CurrentCellColumn	the current column.
CurrentCellRow	the current row.
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
FirstVisibleColumn	the first visible column of the scrollable area of the grid view, fixed columns are ignored.
FirstVisibleRow	the index of the first visible row in the grid.
FrozenColumnCount	the number of columns that are excluded from horizontal scrolling.

Name	Description
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
RowCount	the number of rows.
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
SelectedCells	a list of strings that contains the selected cells, each of which has the format "<row>,<column id>".
SelectedColumns	a list of strings of currently selected columns.
SelectedRows	a comma separated list of row index numbers or index ranges, such as "1,4,6-8".
SelectionMode	the selection mode. Possible values are: "RowsAndColumns", "ListboxSingle", ListboxMultiple", "Free".
Text	The text of the control. (Inherited from *TestObject*)
Title	the title of the grid control.
ToolbarButtonCount	the number of tool bar buttons (separators included).
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VisibleRowCount	the number of visible rows of the grid view.
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/

Name	Description
	Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearSelection	Removes all row, column and cell selections.
Click	Clicks on the specified cell.
ClickCurrentCell	Clicks the current cell.
ContextMenu	Requests the context menu.
CurrentCellMoved	This function notifies the server that another cell was made the current cell. The function must be called whenever the current cell changes.
DeleteRows	Delete the given rows.
DeselectColumn	Removes the specified column from the list of selected columns.
DoubleClick	Double clicks the specified cell.
DoubleClickCurrentCell	Double clicks on the current cell.
DuplicateRows	Duplicates the given range of rows
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCellColor	Returns an identifier for the color of the cell.
GetCellHeight	Gets the height of the cell in pixels.
GetCellIcon	Gets the icon string of the specified cell.
GetCellLeft	Gets the left position of the cell.
GetCellMaxLength	Gets the maximum length of the cell in number of bytes.
GetCellState	Gets the state of the specified cell. Possible values are: Normal, Error, Warning, Info.
GetCellTooltip	Gets the tooltip of the specified cell.
GetCellTop	Gets the top position of the specified cell.
GetCellType	Gets the cell type of the specified cell.

Name	Description
GetCellValue	Gets the value of the specified cell.
GetCellWidth	Gets the width of the specified cell.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetColorInfo	Returns the description for the color of the cell.
GetColumnDataType	Gets the data type of the column.
GetColumnPosition	Gets the position of the specified column.
GetColumnSortType	Gets the sort type of the column. Possible values are: None, Ascending, Descending.
GetColumnTitles	Gets all column titles of the specified column.
GetColumnTooltip	Gets the tool tip text of the specified column.
GetColumnTotalType	Gets the total type of the column. Possible values are: None, Total, Subtotal.
GetDisplayedColumnTitle	Gets the currently displayed title of the specified column.
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetRowTotalLevel	Gets the level of the row.
GetSymbolInfo	Returns the description for the symbol in the cell.
GetToolbarButtonIcon	Gets the name of the icon of the specified toolbar button.
GetToolbarButtonId	Gets the ID of the specified tool bar button.
GetToolbarButtonText	Gets the text of the specified tool bar button.
GetToolbarButtonTooltip	Gets the tooltip of the specified tool bar button
GetToolbarButtonType	Gets the type of the specified toolbar button. Possible values are: "Button", "ButtonAndMenu", "Menu", "Separator", "Group", "CheckBox".
GetToolbarFocusButton	Gets the position of the current focused tool bar button
HasCellF4Help	Returns true if the specified cell has a value help assigned.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)

Name	Description
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
InsertRows	Inserts a range of rows
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsCellChangeable	Returns true if the specified cell is changeable.
IsCellCheckBoxChecked	Returns true if the check box at the specified position is checked.
IsCellHotspot	Returns true if the cell is a link.
IsCellSymbol	Returns true if the text in the cell is displayed in the SAP symbol font.
IsCellTotalExpander	Returns true if the specified cell contains a total expander button.
IsColumnFiltered	Returns true if the specified column is filtered.
IsColumnKey	Returns true if the specified column is a key column.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsToolbarButtonChecked	Returns true if the specified tool bar button is checked.
IsToolbarButtonEnabled	Returns true if the specified button is enabled.
IsTotalRowExpanded	whether the specified total row is expanded.
ModifyCell	Modifies the value of the specified cell.
ModifyCheckBox	Modifies the value of the specified checkbox cell.
MoveRows	Moves the rows with an index greater than or equal to fromRow up to an index less than or equal to toRow to the position of destRow.
PressButton	Presses the button placed in the specified cell.
PressButtonCurrentCell	Clicks a button placed in the current cell.
PressColumnHeader	Clicks on the header of the specified column.
PressEnter	Presses the enter key.
PressF1	Presses the F1 key.
PressF4	Presses the F4 key.
PressToolbarButton	Clicks the specified button in the grid view's tool bar.

Name	Description
PressToolbarContextButton	Opens the context menu of the specified button of the grid view's tool bar.
PressTotalRow	Presses the row button of the specified total row.
PressTotalRowCurrentCell	Presses the expansion button on the current cell.
SelectAll	Selects the whole grid content (i.e. all rows and all columns).
SelectColumn	Adds the specified column to the list of the selected columns.
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SelectionChanged	Notifies the server that the selection has changed
SelectToolbarMenuItem	Selects an item from the context menu of the grid view's tool bar.
SetColumnOrder	Sets the order of the columns.
SetColumnWidth	Sets the width of the specified column.
SetCurrentCell	Sets the specified cell as the current cell.
SetCurrentCellColumn	Specifies the column of the current cell.
SetCurrentCellRow	Sets the row of the current cell.
SetFirstVisibleColumn	Sets the first visible column.
SetFirstVisibleRow	Sets the first visible row.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectedRows	Selects the specified rows.
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TriggerModified	Notifies the server of multiple changes in cells. Typically this method should be called after multiple calls to ModifyCell.

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapHorizontalScrollBar Class

Description
Class for horizontal scrollbars.

Inheritance Hierarchy

- *SapScrollbar*
  - SapHorizontalScrollBar

Syntax

```
'Declaration
Public Class SapHorizontalScrollBar _
Inherits SapScrollbar
```

Properties

Name	Description
Maximum	the maximum scrollbar value. (Inherited from *SapScrollbar*)
Minimum	the minimum scrollbar value. (Inherited from *SapScrollbar*)
PageSize	the scrollbar pagesize. (Inherited from *SapScrollbar*)
Position	the current scrollbar position. (Inherited from *SapScrollbar*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
ScrollTo	Scrolls to the given position. (Inherited from *SapScrollbar*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapHTMLViewer Class

Description
Class for SAP HTMLView.

Inheritance Hierarchy

- *SapShell*
  - SapHTMLViewer

Syntax

```
'Declaration
Public Class SapHTMLViewer _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.

Name	Description
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DocumentComplete	whether the document is complete.
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)
WindowHandle	the window handle of the control that is connected to the GuiShell.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
ContextMenu	Emulates the context menu request. Applies only to context menus provided by the backend.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SapEvent	This function submits an HTML form to the backend.
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a

Name	Description
	WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapLabel Class

Description
SapLabel class is the class for all SAP labels.

Inheritance Hierarchy

- *SapComponent*
  - SapLabel

Syntax
```
'Declaration
Public Class SapLabel _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CharLeft	the left coordinate of the label in character metric.
CharTop	the top coordinate of the label in character metric.
CharWidth	the width of the element in character metric.
ColorIndex	the color index of the label.
ColorInverse	whether the inverse color style is set for the label.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DisplayedText	the text of the label.

Name	Description
Height	the current height of the component. (Inherited from *SapComponent*)
Highlighted	whether the label has the 'highlighted' flag assigned.
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsHotspot	whether a mouse click on the label causes a round trip.
IsLeftLabel	whether the component has the 'assign left' flag assigned.
IsListElement	whether the element is on an ABAP list.
IsRightLabel	whether the component has the 'assign right' flag assigned.
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Maxlength	the maximum text length of the label in code units.
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
Numerical	whether the label may only contain numbers.
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapMenu Class

Description
Class for SAP Menu.

Inheritance Hierarchy

- *SapComponent*
  - SapMenu
    - *SapContextMenu*

Syntax

```
'Declaration
Public Class SapMenu _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)

Name	Description
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Select	Select the menu.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapMenubar Class

Description
Class for SAP MenuBar.

Inheritance Hierarchy

- *SapComponent*
  - SapMenubar

Syntax

```
'Declaration
Public Class SapMenubar _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)

Name	Description
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapNetPlan Class

Description
Class for SAP Net Chart.

Inheritance Hierarchy

- *SapShell*
  - SapNetPlan

Syntax

```
'Declaration
Public Class SapNetPlan _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityDescription	the accessibility description of the shell.
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
LinkCount	the number of links in the net.

Name	Description
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
NodeCount	the number of Nodes in the net.
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)
WindowHandle	the window handle of the control that is connected to the GuiShell.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetLinkContent	Returns the content of the link.
GetNodeContent	Returns the content of the node.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SendData	This function emulates the output of each action triggered at the control side. The result of the action is sent to the server.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapOfficeIntegration Class

Description
Class for SAP OfficeIntegration.

Inheritance Hierarchy

- *SapShell*
  - SapOfficeIntegration

Syntax

```
'Declaration
Public Class SapOfficeIntegration _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapOkCodeField Class

Description
SapOkCodeField class is the class for sap OkCodeField.

Inheritance Hierarchy

- *SapComponent*
  - SapOkCodeField

Syntax
```
'Declaration
Public Class SapOkCodeField _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)

Name	Description
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
Opened	whether the GuiOkCodeField is collapsed.
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Close	Collapses the GuiOkCodeField.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Open	Expands the GuiOkCodeField.
PressF1	Emulate pressing the F1 key while the focus is on the GuiOkCodeField.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetText	Sets the text of the control to the specified text.
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapPicture Class

Description
Class for SAP picture.

Inheritance Hierarchy

- *SapShell*
  - SapPicture

Syntax

```
'Declaration
Public Class SapPicture _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DisplayMode	the display mode of the picture, i.e. "Normal", "Stretch", "Fit", "NormalCenter", "FitCenter"
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
Height	the current height of the component. (Inherited from *SapComponent*)
Icon	the SAPGUI icon code e.g. ("@01@")
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)

Name	Description
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Url	the URL of the picture
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	clicks on a picture
ClickControlArea	clicks on the specified position
ClickPictureArea	clicks on the specified position in the picture area
ContextMenu	The function opens a context menu on the given position. The coordinates should be given in pixels with respect to the picture control as it is displayed on the screen.
DoubleClick	double clicks on the picture
DoubleClickControlArea	double clicks on the specified position
DoubleClickPictureArea	double clicks on the specified position in the picture area

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the

Name	Description
	timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapRadioButton Class

Description
SapRadioButton class is the class for radio button controls.

Inheritance Hierarchy

- *SapComponent*
  - SapRadioButton

Syntax
```
'Declaration
Public Class SapRadioButton _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CharHeight	the height of the element in character metric
CharLeft	the left coordinate of the element in character metric
CharTop	the top coordinate of the element in character metric
CharWidth	the width of the element in character metric
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Flushing	whether the value change causes a round trip
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsLeftLabel	whether the component has the 'assign left' flag

Name	Description
IsRightLabel	whether the component has the 'assign right' flag
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Selected	whether the radio button is selected
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Select	Selects the radio button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapScrollbar Class

Description
Class for SAP Scrollbar.

Inheritance Hierarchy

- *TestObject*
  - SapScrollbar
    - *SapHorizontalScrollBar*

- *SapVerticalScrollBar*

Syntax

```
'Declaration
Public Class SapScrollbar _
Inherits TestObject
```

Properties

Name	Description
Maximum	the maximum scrollbar value.
Minimum	the minimum scrollbar value.
PageSize	the scrollbar pagesize.
Position	the current scrollbar position.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
ScrollTo	Scrolls to the given position.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapScrollContainer Class

Description
Class for SAP UserArea.

Inheritance Hierarchy

- *SapComponent*
  - SapScrollContainer

Syntax

```
'Declaration
Public Class SapScrollContainer _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)

Name	Description
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapShell Class

Description
Class for SAP SapShell.

Inheritance Hierarchy

- *SapComponent*
  - SapShell
    - *SapBarChart*
    - *SapCalendar*
    - *SapChart*
    - *SapColorSelector*
    - *SapGridView*
    - *SapHTMLViewer*
    - *SapNetPlan*
    - *SapOfficeIntegration*
    - *SapPicture*
    - *SapTextEdit*
    - *SapToolbarControl*
    - *SapTree*

Syntax

```
'Declaration
Public Class SapShell _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations.
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)

Name	Description
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapSimpleContainer Class

Description
Class for SAP SimpleContainer.

Inheritance Hierarchy

- *SapComponent*
  - SapSimpleContainer

Syntax

```
'Declaration
Public Class SapSimpleContainer _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapSplitterContainer Class

Description
Class for SAP splitter container.

Inheritance Hierarchy

- *SapComponent*
  - SapSplitterContainer

Syntax

```
'Declaration
Public Class SapSplitterContainer _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)

Name	Description
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapStatusbar Class

Description
Class for SAP statusbar.

Inheritance Hierarchy

- *SapComponent*
  - SapStatusbar

Syntax

```
'Declaration
Public Class SapStatusbar _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)

Name	Description
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
MessageId	the name of the message class used in the ABAP message call
MessageNumber	the message number used in the ABAP message call
MessageType	the type of the displayed message in the statusbar, i.e. S (Success), W (Warning), E (Error), A (Abort), I (Information)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha

Name	Description
	channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
DoubleClick	double-clicks the statusbar
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapTab Class

Description
Class for SAP Tab.

Inheritance Hierarchy

- *SapComponent*
  - SapTab

Syntax

```
'Declaration
Public Class SapTab _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)

Name	Description
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Select	Selects the tab
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a

Name	Description
	timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapTable Class

Description
SapTable class is the class for SAP tables.

Inheritance Hierarchy

- *SapComponent*
  - SapTable

Syntax
```
'Declaration
Public Class SapTable _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)

Name	Description
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CharHeight	the height of the element in character metric.
CharLeft	the left coordinate of the element in character metric.
CharTop	the top coordinate of the element in character metric.
CharWidth	the width of the element in character metric.
ColumnCount	the number of columns in the table.
ColumnSelectionMode	the column selection mode
CurrentColumn	the index of the current column.
CurrentRow	the index of the current row.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
RowCount	the number of rows in the table including invisible rows.
RowSelectionMode	the row selection mode.
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
TableFieldName	the table field name.
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VisibleRowCount	the number of visible rows in the table.
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ConfigureLayout	Opens the configuration layout dialog, in which the layout of the table can be changed.
DeselectAllColumns	This function can be used for table controls with a button that allows to de-select all columns in one step.
DeselectColumn	Deselects the given column.
DeselectRow	Deselects the given row.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCell	Returns a visible table cell.
GetCellAbsolute	Returns a table cell. If the row is not visible, it will be scrolled into view.
GetCellsForRow	Returns the cells for the given row.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetColumnTitle	This method returns the title of the given column.

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVisibleCellsForColumn	Returns the visible cells for the given column.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsColumnSelected	Returns true if the given column is selected.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsRowSelected	Returns true if the given row is selected.
ReorderTable	This method re-orders the columns.
SelectAllColumns	This function can be used for table controls with a button that allows to select all columns in one step.
SelectColumn	Selects the given column.
SelectRow	Selects the given row.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapTabStrip Class

Description
Class for SAP tab pane.

Inheritance Hierarchy

- *SapComponent*
  - SapTabStrip

Syntax

```
'Declaration
Public Class SapTabStrip _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
SelectedTab	the current selected tab.
Tabs	all tabs of the tab pane.
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)

Name	Description
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown.

Name	Description
	The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapTextEdit Class

Description
Class for SAP TextEdit.

Inheritance Hierarchy

- *SapShell*
  - SapTextEdit

Syntax

```
'Declaration
Public Class SapTextEdit _
Inherits SapShell
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentColumn	the column index in which the caret is currently positioned.
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
CurrentLine	the line index in which the caret is currently positioned.

Name	Description
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
FirstVisibleLine	the first visible line which is visualized at the top border of the text edit control.
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
LastVisibleLine	the last visible line which is visualized at the bottom border of the text edit control.
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
LineCount	the number of all lines in the text edit control.
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
SelectedText	the currently selected text.
SelectionEndColumn	the column index in which the selection ends.
SelectionEndLine	the line index in which the selection ends.
SelectionIndexEnd	the overall end index of the text selection.
SelectionIndexStart	the overall start index of the text selection.
SelectionStartColumn	the column start index of the text selection.
SelectionStartLine	the line index in which the selection starts.
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ContextMenu	Requests the context menu.
DoubleClick	Double clicks the text edit control.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetLineText	Returns the text of the specified line.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsBreakpointLine	whether the specified line contains a breakpoint.
IsCommentLine	Returns true if the specified line is a comment line.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsHighlightedLine	Returns true if the specified line is highlighted.
IsProtectedLine	Returns true if the specified line is protected.
PressF1	Presses the F1 key.
PressF4	Presses the F4 key.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionIndexes	Selects the text within the specified range.
SetText	Sets the specified text in the text edit control.
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapTextField Class

Description
SapTextField is the class for single-line and multi-line fields whose text can be modified by the user.

Inheritance Hierarchy

- *SapComponent*
  - SapTextField

Syntax
```
'Declaration
Public Class SapTextField _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)

Name	Description
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DisplayedText	the text as it is displayed on the screen, including preceding or trailing blanks.
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsPassword	whether the text field is a password text field.
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetPasswordText	Sends an encrypted password to the object.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetText	Sets the text of the control to the specified text.
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapTitlebar Class

Description
Class for SAP Titlebar.

Inheritance Hierarchy

- *SapComponent*
  - SapTitlebar

Syntax

```
'Declaration
Public Class SapTitlebar _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)

Name	Description
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapToolbar Class

Description
Class for SAP Toolbar.

Inheritance Hierarchy

- *SapComponent*
  - SapToolbar

Syntax

```
'Declaration
Public Class SapToolbar _
Inherits SapComponent
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapToolbarControl Class

Description
Class for SAP ToolbarControl.

Inheritance Hierarchy

- *SapShell*
  - SapToolbarControl

Syntax

```
'Declaration
Public Class SapToolbarControl _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
ButtonCount	the number of toolbar buttons including separators.
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)
FocusedButton	the zero-based index of the button that currently has the focus.
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)

Name	Description
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)
WindowHandle	the window handle of the control that is connected to the GuiShell.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetButtonIcon	the name of the icon of the specified toolbar button.

Name	Description
GetButtonType	the type of the specified toolbar button. Possible values are: "Button", "ButtonAndMenu", "Menu", "Separator", "Group", "CheckBox"
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsButtonChecked	if the button is currently checked (pressed).
IsButtonEnabled	if the button can be pressed.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
PressContextButton	Press the context button with the given id.
SelectButton	emulates pressing the button with the given id.
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)

Name	Description
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SelectMenuItem	emulates selecting the menu item with the given id.
SelectMenuItemByText	emulates selecting the menu item by menu item text.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapTree Class

Description
SapTree class is the class for all SAP trees. Some of the methods that have Get as a prefix require the nodeKey and the ItemName as iput parameters of type string. These parameters are different for each individual SapTree element. To obtain the value of these parameters, record an action on the desired item and use the parameters that you have recorded.

Inheritance Hierarchy

- *SapShell*
  - SapTree

Syntax

```
'Declaration
Public Class SapTree _
Inherits SapShell _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityDescription	the accessibility description of the shell.
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
ColumnHeaders	a collection of the titles of the columns.
ColumnNames	a collection of the column names.
ColumnOrder	the column order.
ColumnTitles	a collection of the titles of the columns.
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
DragDropSupported	whether the shell allows drag and drop operations. (Inherited from *SapShell*)

Name	Description
FocusedNodeKey	the key of the focused node.
Height	the current height of the component. (Inherited from *SapComponent*)
HierarchyHeaderWidth	the width of the hierarchy header in pixels.
HierarchyTitle	the hierarchy title.
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
NodeKeys	a collection containing either the node key of the root node, or the node keys of all nodes in the tree, depending on the application.
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
SelectedItemColumn	the key of the currently selected item column.
SelectedItemNode	the key of the currently selected item node.
SelectedNode	the key of the currently selected node.
SelectedNodes	a collection containing the node keys of all the selected nodes in the tree.
SelectionMode	the selection mode.
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
TopNode	the key of the top node.
TreeType	the tree type. (simple tree, list tree or column tree)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ChangeCheckbox	This method emulates changing a checkbox state.
CollapseNode	This function closes the node with the key nodeKey.
DefaultContextMenu	This method requests a context menu for the whole Tree Control.
DoubleSelectItem	This function emulates double clicking on a text item.
DoubleSelectNode	This function emulates double clicking a node.
EnsureVisibleHorizontalItem	This function scrolls the Tree horizontally until the Item is visible.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
ExpandNode	This function expands the node with the key nodeKey.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
FindNodeKeyByPath	Return the node key for the given path (e.g. 2\1\2).
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetAbapImage	Gets the abap image.
GetCheckBoxState	Retrieves the CheckBox state (true = Checked, false = Unchecked).
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetColumnIndexFromName	Gets the column index for the given name.
GetColumnKeys	The keys of all the items in the given column.
GetColumnTitleFromName	Gets the column title for the given name.
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHierarchyLevel	Gets the hierarchy level for the given key.

Name	Description
GetItemHeight	Returns the current item height.
GetItemLeft	The left position of the component.
GetItemStyle	The style of the given item.
GetItemText	For multi column trees, this function returns the text of the item specified by the key and name parameters.
GetItemTextColor	Returns the item text color.
GetItemTooltip	Returns the item tooltip text.
GetItemTop	The top position of the component.
GetItemType	Returns the type of the given node.
GetItemWidth	Returns the current item width.
GetListTreeNodeItemCount	Returns the node item count.
GetNextNodeKey	Returns the key of the next node.
GetNodeAbapImage	Returns the abap image for the given node.
GetNodeChildrenCount	This function returns the number of children of the node given.
GetNodeChildrenCountByPath	This function returns the number of children of the node given by the path parameter (e.g. 2\1\2).
GetNodeHeight	The current height of the node.
GetNodeIndex	The index of the given node.
GetNodeItemHeaders	Returns the headers for the given node.
GetNodeLeft	The left position of the component.
GetNodePathByKey	Given a node key, the path is retrieved (e.g. 2\1\2).
GetNodeStyle	Returns the style of the node.
GetNodeText	This function returns the text of the node specified by the given key.
GetNodeTextByPath	The text of a node defined by the given path is returned.
GetNodeTextColor	Returns the node text color.
GetNodeTooltip	Returns the node tooltip text.
GetNodeTop	The top position of the component.
GetNodeWidth	Returns the current node width.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetParentNodeKey	Key of the parent node of the node specified by the given key.
GetPreviousNodeKey	Key of the previous node of the node specified by the given key.
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetStyleDescription	Returns the description for the given style.
GetSubNodesKeys	Collection of the keys of all subnodes of the node specified by the given key.
HeaderContextMenu	This method requests a context menu for a header.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsFolder	Whether the given node is a folder.
IsFolderExpandable	Whether the folder is expandable.
IsFolderExpanded	Whether the folder is expanded.
IsHighLighted	Whether the node is highlighted.
ItemContextMenu	This method requests a context menu for an item.
NodeContextMenu	This method requests a context menu for a node.
PressKey	This method emulates pressing a key.
SelectButton	This method emulates pressing a button.
SelectColumn	This function adds a column to the column selection. A node or item selection is removed.
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)

Name	Description
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SelectHeader	This method emulates clicking a header.
SelectItem	This function emulates the selection of an item. This selection removes all other selections.
SelectLink	This function emulates triggering a link.
SelectNode	The node with the key nodeKey is added to the Node Selection.
SetCheckBoxState	Changes the state of the checkbox in the given node.
SetColumnOrder	Sets the order of the tree's column.
SetColumnWidth	This function sets the width of a column in pixels.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetHierarchyHeaderWidth	Sets the width of the Hierarchy Header in pixels.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectedNode	Select the given node.
SetTopNode	Sets the given key as top node.
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
UnselectAll	All selections are removed.
UnselectColumn	This function removes a column from the column selection.
UnselectNode	The node with the key nodeKey is removed from the Node Selection.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapUserArea Class

Description
Class for SAP UserArea.

Inheritance Hierarchy

- *SapComponent*
  - SapUserArea

Syntax

```
'Declaration
Public Class SapUserArea _
Inherits SapComponent _
Implements ISapContextMenuable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
CurrentContextMenu	The current context menu. This is only set when a context menu is available at the shell object.

Name	Description
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
SelectContextMenuItem	Select an item from the controls context menu. (Inherited from *ISapContextMenuable*)
SelectContextMenuItemByPosition	This method allows you to select a context menu item using the position of the item. (Inherited from *ISapContextMenuable*)

Name	Description
SelectContextMenuItemByText	Select a menu item of a context menu using the text of the item and possible higher level menus. (Inherited from *ISapContextMenuable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapVerticalScrollBar Class

Description
Class for vertical scrollbars.

Inheritance Hierarchy

- *SapScrollbar*
  - SapVerticalScrollBar

Syntax
```
'Declaration
Public Class SapVerticalScrollBar _
Inherits SapScrollbar
```

Properties

Name	Description
Maximum	the maximum scrollbar value. (Inherited from *SapScrollbar*)
Minimum	the minimum scrollbar value. (Inherited from *SapScrollbar*)
PageSize	the scrollbar pagesize. (Inherited from *SapScrollbar*)
Position	the current scrollbar position. (Inherited from *SapScrollbar*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
ScrollTo	Scrolls to the given position. (Inherited from *SapScrollbar*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SapWindow Class

Description
SapWindow class is the class for a top-level window.

Inheritance Hierarchy
- *SapComponent*
  - SapWindow

Syntax
```
'Declaration
Public Class SapWindow _
Inherits SapComponent _
Implements IMoveable
```

Properties

Name	Description
AccessibilityText	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTextOnRequest	an additional text for accessibility support. (Inherited from *SapComponent*)
AccessibilityTooltip	an additional tooltip text for accessibility support. (Inherited from *SapComponent*)
ButtonbarVisible	whether the application toolbar, the lower toolbar within SAP GUI, is visible.
Changeable	whether an object is changeable. (Inherited from *SapComponent*)
DefaultTooltip	the tooltip text which is generated from the short text defined in the data dictionary for the given screen element type. (Inherited from *SapComponent*)
GuiFocus	the element within SAP GUI that currently has the focus.
Height	the current height of the component. (Inherited from *SapComponent*)
IconName	whether the object has been assigned an icon, then this property is the name of the icon, otherwise it is an empty string. (Inherited from *SapComponent*)
IsSymbolFont	whether the component's text is visualized in the SAP symbol font. (Inherited from *SapComponent*)
Left	the left position of the element in screen coordinates. (Inherited from *SapComponent*)
Modified	whether the state has been changed by the user and this change has not yet been sent to the SAP system. (Inherited from *SapComponent*)
Name	the name of the component. (Inherited from *SapComponent*)
ScreenLeft	the left position of the component in screen coordinates. (Inherited from *SapComponent*)
ScreenTop	the top position of the component in screen coordinates. (Inherited from *SapComponent*)
StatusbarVisible	whether the statusbar is visible.
SystemFocus	the element within SAP GUI that has the focus from the server application's perspective.
Text	The text of the control. (Inherited from *TestObject*)
TitlebarVisible	whether the titlebar is visible.
ToolbarVisible	whether the toolbar is visible.
Tooltip	the tooltip text. (Inherited from *SapComponent*)
Top	the top coordinate of the element in screen coordinates. (Inherited from *SapComponent*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	the current width of the component. (Inherited from *SapComponent*)
WindowHandle	the window handle of the control that is connected to the GuiShell.
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)
WorkingPaneHeight	the current height of the working pane.
WorkingPaneWidth	the current width of the working pane.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)

Name	Description
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsVkeyAllowed	The method returns True if the given virtual key can be executed on the current screen.
JumpBackward	Execute the Ctrl+Shift+Tab key on the window to jump backward one block.
JumpForward	Execute the Ctrl+Tab key on the window to jump forward one block.
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
Move	Moves the window. (Inherited from *IMoveable*)
ResizeWorkingPane	The ResizeWorkingPane function will resize the window so that the available working area has the given width and height in character metric.

Name	Description
ResizeWorkingPaneEx	The ResizeWorkingPaneEx function will resize the window so that the available working area has the given width and height in pixels.
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
SendVKey	Send a virtual key to the system.
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
ShowContextMenu	Shows the context menu of the control. (Inherited from *SapComponent*)
Size	Resizes the window. (Inherited from *IMoveable*)
TabBackward	Execute the Shift+Tab key on the window to jump backward one element.
TabForward	Execute the Tab key on the window to jump forward one element.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Silverlight Class Reference

Lists the available classes for testing Silverlight controls.

SLApplication Class

Description

The container for Silverlight controls.

Inheritance Hierarchy

- *SLBase*
  - SLApplication

Syntax

```
'Declaration
Public Class SLApplication _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLAutoCompleteBox Class

Description

Represents a control that provides a text box for user input and a drop-down that contains possible matches based on the input in the text box.

Inheritance Hierarchy

- *SLBase*
  - SLAutoCompleteBox

Syntax

```
'Declaration
Public Class SLAutoCompleteBox _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
HorizontallyScrollable	a value that indicates whether the combo box can scroll horizontally.
HorizontalScrollPercent	the current horizontal scroll position or negative one (-1) if there is no valid scroll position.
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsOpen	a value that indicates whether the auto complete box is open
Items	a list containing items of the current matches. If nothing was entered in the control's text box an empty list is returned.
ItemTexts	a list containing the texts of the current matches. If nothing was entered in the control's text box an empty list is returned.
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticallyScrollable	a value that indicates whether the combo box can scroll vertically.
VerticalScrollPercent	the current vertical scroll position or negative one (-1) if there is no valid scroll position.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the drop-down list that contains the matches if it is currently open.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollHorizontal	Scrolls the currently visible region of the content area, horizontally, the specified ScrollAmount.
ScrollVertical	Scrolls the currently visible region of the content area, vertically, the specified ScrollAmount.
Select	Types the given text into the auto box complete box and selects a match.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetHorizontalScrollPercent	Sets the horizontal scroll position as a percentage of the total content area within the combo box.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetVerticalScrollPercent	Sets the vertical scroll position as a percentage of the total content area within the combo box.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLBase Class

Description
Provides a Silverlight framework-level set of properties, events, and methods for Silverlight elements.

Inheritance Hierarchy

- *TestObject*
  - SLBase
    - *SLApplication*
    - *SLAutoCompleteBox*
    - *SLButton*
    - *SLCalendar*
    - *SLCalendarButton*
    - *SLCalendarDayButton*
    - *SLCheckBox*
    - *SLComboBox*
    - *SLComboBoxItem*
    - *SLDataGrid*
    - *SLDataGridCell*
    - *SLDataGridDetails*
    - *SLDataGridRow*

- *SLDataPager*
- *SLDatePicker*
- *SLDescriptionViewer*
- *SLFrame*
- *SLGridSplitter*
- *SLGroup*
- *SLHeader*
- *SLHeaderItem*
- *SLHorizontalScrollBar*
- *SLHyperlinkButton*
- *SLImage*
- *SLListBox*
- *SLListItem*
- *SLMediaElement*
- *SLMenu*
- *SLMenuBar*
- *SLMenuItem*
- *SLMultiScaleImage*
- *SLPane*
- *SLPasswordBox*
- *SLPopup*
- *SLProgressBar*
- *SLRadioButton*
- *SLRepeatButton*
- *SLRichTextBox*
- *SLSeparator*
- *SLSlider*
- *SLSpinner*
- *SLSplitButton*
- *SLStatusBar*
- *SLTabControl*
- *SLTabItem*
- *SLTable*
- *SLTextBlock*
- *SLTextBox*
- *SLThumb*
- *SLTitleBar*
- *SLToggleButton*
- *SLToolBar*
- *SLToolTip*
- *SLTreeView*
- *SLTreeViewItem*
- *SLValidationSummary*
- *SLVerticalScrollBar*
- *SLWindow*

Syntax

```
'Declaration
Public Class SLBase _
```

```
Inherits TestObject _
Implements IClickable, IFocusable, IKeyable
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element.
ClassName	the simple class name of the element.
IsEnabled	a value that indicates whether the element is enabled.
Name	a string containing the UI Automation name for the element.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type.
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLButton Class

Description
Identifies a button control.

Inheritance Hierarchy

- *SLBase*
  - SLButton

Syntax
```
'Declaration
Public Class SLButton _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Clicks the button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLCalendar Class

Description
Identifies a calendar control.

Inheritance Hierarchy

- *SLBase*
  - SLCalendar

Syntax

```
'Declaration
Public Class SLCalendar _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
CanSelectMultiple	whether multiple dates can be selected
ClassName	the simple class name of the element. (Inherited from *SLBase*)
DisplayMode	a value that indicates whether the calendar displays a month, year, or decade.
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsSelectionRequired	whether at least one date has to be selected
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
SelectedDate	the currently selected date in the currently displayed view.
SelectedDates	a list of selected dates in the currently displayed view.

Name	Description
SupportedDisplayModes	a list of supported display modes.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SelectDate	Selects the given date.
SelectDates	Selects the given dates.
SelectRange	Selects a range of dates.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SwitchToDecadeView	Switches the calendar display mode to 'decade', if supported.

Name	Description
SwitchToMonthView	Switches the calendar display mode to 'month', if supported.
SwitchToYearView	Switches the calendar display mode to 'year', if supported.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLCalendarButton Class

Description
Identifies a calendar button.

Inheritance Hierarchy

- *SLBase*
  - SLCalendarButton

Syntax

```
'Declaration
Public Class SLCalendarButton _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsSelected	a value that indicates whether the calendar button is selected
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Clicks the button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLCalendarDayButton Class

Description
Identifies a calendar day button.

Inheritance Hierarchy

- *SLBase*
  - SLCalendarDayButton

Syntax

```
'Declaration
Public Class SLCalendarDayButton _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsSelected	a value that indicates whether the calendar day button is selected
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Clicks the button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLCheckBox Class

Description
Identifies a check box control.

Inheritance Hierarchy

- *SLBase*
  - SLCheckBox

Syntax

```
'Declaration
Public Class SLCheckBox _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
ToggleState	the toggle state of the check box
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetIndeterminate	Set the check box to the indeterminate state.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Toggle	Cycles through the toggle states of the check box in this order: On, Off and, if supported, Indeterminate.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLComboBox Class

Description

Identifies a combo box control.

Combo boxes in Silverlight require to be opened once before information about their items can be retrieved.

Inheritance Hierarchy

- *SLBase*
  - SLComboBox

Syntax

```
'Declaration
Public Class SLComboBox _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
HorizontallyScrollable	a value that indicates whether the combo box can scroll horizontally.
HorizontalScrollPercent	the current horizontal scroll position or negative one (-1) if there is no valid scroll position.
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsOpen	a value that indicates whether the combo box is open
Items	a complete list of items.
ItemTexts	a complete list of item texts.
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
SelectedIndex	the index of the selected item or negative one (-1) if the selection is empty.
SelectedItem	the selected item or `Nothing` if the selection is empty.

Name	Description
SelectedItemText	the text of the selected item or an empty string if the selection is empty.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticallyScrollable	a value that indicates whether the combo box can scroll vertically.
VerticalScrollPercent	the current vertical scroll position or negative one (-1) if there is no valid scroll position.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the combo box.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Open	Opens the combo box.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollHorizontal	Scrolls the currently visible region of the content area, horizontally, the specified ScrollAmount.

Name	Description
ScrollVertical	Scrolls the currently visible region of the content area, vertically, the specified ScrollAmount.
Select	Selects an item in the combo box.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetHorizontalScrollPercent	Sets the horizontal scroll position as a percentage of the total content area within the combo box.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetVerticalScrollPercent	Sets the vertical scroll position as a percentage of the total content area within the combo box.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLComboBoxItem Class

Description
Identifies a combo box item control.

Inheritance Hierarchy

- *SLBase*
  - SLComboBoxItem

Syntax

```
'Declaration
Public Class SLComboBoxItem _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsSelected	a value that indicates whether the combo box item is selected
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects the combo box item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLDataGrid Class

Description

Identifies a data grid control.

If the data grid has a vertical scroll bar, rows which are not in the visible area of the data grid do not expose any information (e.g. text, caption). In order to interact with such a row or retrieve information about the row it needs to be scrolled into view first.

Inheritance Hierarchy

- *SLBase*
  - SLDataGrid

Syntax

```
'Declaration
Public Class SLDataGrid _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)

Name	Description
CanSelectMultiple	whether the data grid allows more than one row to be selected concurrently.
ClassName	the simple class name of the element. (Inherited from *SLBase*)
ColumnCount	the number of columns in the data grid
ColumnHeaders	a list containing the header item for every column
ColumnHeaderTexts	a list containing the text of the header item for every column
HorizontallyScrollable	a value that indicates whether the data grid can scroll horizontally.
HorizontalScrollPercent	the current horizontal scroll position or negative one (-1) if there is no valid scroll position.
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
RowCount	the number of rows in the data grid
Rows	a list of rows in the data grid
RowTexts	a list of row texts in the data grid. The text for every row is returned as a string that contains the text of every cell separated with a blank character.
SelectedCell	the currently selected cell or `Nothing` if no cell is selected
SelectedCellText	the text of the currently selected cell or an empty string if no cell is selected
SelectedRow	the currently selected row or `Nothing` if no row is selected
SelectedRows	a list of currently selected rows or an empty list if no row is selected
SelectedRowText	the text of the currently selected row or an empty string if no row is selected
SelectedRowTexts	the text of the currently selected rows or an empty list if no row is selected
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticallyScrollable	a value that indicates whether the data grid can scroll vertically.
VerticalScrollPercent	the current vertical scroll position or negative one (-1) if there is no valid scroll position.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollHorizontal	Scrolls the currently visible region of the content area, horizontally, the specified ScrollAmount.
ScrollIntoView	Scrolls the specified row or cell into view.
ScrollVertical	Scrolls the currently visible region of the content area, vertically, the specified ScrollAmount.
Select	Selects the specified row.
SelectCell	Selects the specified cell.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetHorizontalScrollPercent	Sets the horizontal scroll position as a percentage of the total content area within the data grid.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetVerticalScrollPercent	Sets the vertical scroll position as a percentage of the total content area within the data grid.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLDataGridCell Class

Description
Identifies a cell within a row of a data grid control.

Inheritance Hierarchy

- *SLBase*
  - SLDataGridCell

Syntax

```
'Declaration
Public Class SLDataGridCell _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsSelected	a value that indicates whether the cell is selected
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scroll the cell into view.
Select	Selects the cell
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the

Name	Description
	timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLDataGridDetails Class

Description
Identifies a control that displays additional details within a data grid row.

Inheritance Hierarchy

- *SLBase*
  - SLDataGridDetails

Syntax

```
'Declaration
Public Class SLDataGridDetails _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLDataGridRow Class

Description
Identifies a row in data grid.

Inheritance Hierarchy

- *SLBase*
  - SLDataGridRow

Syntax

```
'Declaration
Public Class SLDataGridRow _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
Cells	the cells of the data grid row.
CellTexts	the text of the cells of the data grid row.
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsSelected	a value that indicates whether the row is selected.
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)

Name	Description
RowHeader	the header for this row
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the row into view.
Select	Selects the row
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLDataPager Class

Description
Identifies a data pager for paging through a collection of data.

Inheritance Hierarchy

- *SLBase*

- SLDataPager

Syntax

```
'Declaration
Public Class SLDataPager _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
CurrentPage	The current page.
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
PageCount	The count of the pages.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MoveToFirstPage	Moves the data pager to the first page.
MoveToLastPage	Moves the data pager to the last page.

Name	Description
MoveToNextPage	Moves the data pager to the next page.
MoveToPage	Moves the data pager to the given page.
MoveToPreviousPage	Moves the data pager to the previous page.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object,

Name	Description
	for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLDatePicker Class

Description

Represents a control that allows the user to select a date.

Inheritance Hierarchy

- *SLBase*
  - SLDatePicker

Syntax

```
'Declaration
Public Class SLDatePicker _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsCalendarOpen	a value whether the drop-down Calendar is open.
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
SelectedDate	the selected date or `Nothing` if no date is selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
CloseCalendar	Closes the drop-down Calendar.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenCalendar	Opens the drop-down Calendar.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLDescriptionViewer Class

Description
Identifies a description viewer control.

Inheritance Hierarchy

- *SLBase*
  - SLDescriptionViewer

Syntax

```
'Declaration
Public Class SLDescriptionViewer _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLFrame Class

Description
Identifies a frame control.

Inheritance Hierarchy

- *SLBase*
  - SLFrame

Syntax
```
'Declaration
Public Class SLFrame _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object,

Name	Description
	for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLGridSplitter Class

Description
Identifies a grid splitter control.

Inheritance Hierarchy

- *SLBase*
  - SLGridSplitter

Syntax

```
'Declaration
Public Class SLGridSplitter _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MoveHorizontal	Moves the splitter horizontally.
MoveVertical	Moves the splitter vertically.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLGroup Class

Description
Identifies a group control, which acts as a container for other controls.

Inheritance Hierarchy

- *SLBase*
  - SLGroup

Syntax

```
'Declaration
Public Class SLGroup _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLHeader Class

Description
Identifies a control that provides a visual container for the labels for rows or columns.

Inheritance Hierarchy

- *SLBase*
  - SLHeader

Syntax
```
'Declaration
Public Class SLHeader _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
HeaderItems	a list containing the header items of this header control
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown.

Name	Description
	The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLHeaderItem Class

Description
Identifies a control that provides a visual label for a row or column.

Inheritance Hierarchy

- *SLBase*
  - SLHeaderItem

Syntax

```
'Declaration
Public Class SLHeaderItem _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects the header item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLHorizontalScrollBar Class

Description
Identifies a horizontal scroll bar.

Inheritance Hierarchy

- *SLBase*
  - SLHorizontalScrollBar

Syntax

```
'Declaration
Public Class SLHorizontalScrollBar _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Maximum	the maximum scroll position
Minimum	the minimum scroll position
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
ScrollPosition	the current scroll position
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Scrolls the scroll bar by the specified amount
ScrollToMaximum	Scrolls the scroll bar to the right end.
ScrollToMinimum	Scrolls the scroll bar to the left end.
ScrollToPosition	Scrolls the scroll bar to the specific position
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLHyperlinkButton Class

Description
Identifies a hyperlink button control.

Inheritance Hierarchy

- *SLBase*
  - SLHyperlinkButton

Syntax

```
'Declaration
Public Class SLHyperlinkButton _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Clicks the hyperlink button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLImage Class

Description
Identifies an image control.

Inheritance Hierarchy

- *SLBase*
  - SLImage

Syntax

```
'Declaration
Public Class SLImage _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)

Name	Description
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLListBox Class

Description
Identifies a list box control.

Inheritance Hierarchy

- *SLBase*
  - SLListBox

Syntax

```
'Declaration
Public Class SLListBox _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
CanSelectMultiple	whether the list box allows more than one child element to be selected concurrently.
ClassName	the simple class name of the element. (Inherited from *SLBase*)
HorizontallyScrollable	a value that indicates whether the list box can scroll horizontally.
HorizontalScrollPercent	the current horizontal scroll position or negative one (-1) if there is no valid scroll position.
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Items	the list of list box items.
ItemTexts	the list of all item texts.
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
SelectedIndex	the index of the first item in the current selection or returns negative one (-1) if the selection is empty.
SelectedIndices	the indices of the currently selected items. An empty list is returned if no items are currently selected.
SelectedItem	the first item in the current selection or returns `Nothing` if the selection is empty
SelectedItems	the currently selected items. An empty list is returned, if no items are currently selected.
SelectedItemText	the text of the first selected item.
SelectedItemTexts	the texts of the currently selected items. An empty list is returned if no items are currently selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticallyScrollable	a value that indicates whether the list box can scroll vertically.
VerticalScrollPercent	the current vertical scroll position or negative one (-1) if there is no valid scroll position.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DeselectAll	Deselects all list items.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)

Name	Description
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollHorizontal	Scrolls the currently visible region of the content area, horizontally, the specified ScrollAmount.
ScrollVertical	Scrolls the currently visible region of the content area, vertically, the specified ScrollAmount.
Select	Selects the given list box item.
SelectAll	Selects all items in the list.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetHorizontalScrollPercent	Sets the horizontal scroll position as a percentage of the total content area within the list box.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetVerticalScrollPercent	Sets the vertical scroll position as a percentage of the total content area within the list box.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLListItem Class

Description
Identifies a list item control.

Inheritance Hierarchy

- *SLBase*

- SLListItem

Syntax

```
'Declaration
Public Class SLListItem _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsSelected	a value that indicates whether the list item is selected
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Deselect	Deselects the item.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scroll the list item into view.
Select	Selects the list item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLMediaElement Class

Description
Represents a control that contains audio and/or video.

Inheritance Hierarchy

- *SLBase*
  - SLMediaElement

Syntax

```
'Declaration
Public Class SLMediaElement _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLMenu Class

Description
Identifies a menu control.

Inheritance Hierarchy

- *SLBase*
  - SLMenu

Syntax

```
'Declaration
Public Class SLMenu _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as

Name	Description
	children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLMenuBar Class

Description
Identifies a menu bar control.

Inheritance Hierarchy

- *SLBase*
  - SLMenuBar

Syntax

```
'Declaration
Public Class SLMenuBar _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLMenuItem Class

Description
Identifies a menu item control.

Inheritance Hierarchy

- *SLBase*
  - SLMenuItem

Syntax
```
'Declaration
Public Class SLMenuItem _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLMultiScaleImage Class

Description
Enables users to open a multi-resolution image, which can be zoomed in on and panned across.

Inheritance Hierarchy

- *SLBase*
  - SLMultiScaleImage

Syntax

```
'Declaration
Public Class SLMultiScaleImage _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLPane Class

Description
Identifies a pane that can optionally have scroll bars.

Inheritance Hierarchy

- *SLBase*
  - SLPane

Syntax

```
'Declaration
Public Class SLPane _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
HorizontallyScrollable	a value that indicates whether the pane can scroll horizontally.
HorizontalScrollPercent	the current horizontal scroll position or negative one (-1) if there is no valid scroll position.
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticallyScrollable	a value that indicates whether the pane can scroll vertically.
VerticalScrollPercent	the current vertical scroll position or negative one (-1) if there is no valid scroll position.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollHorizontal	Scrolls the currently visible region of the content area, horizontally, the specified ScrollAmount.
ScrollVertical	Scrolls the currently visible region of the content area, vertically, the specified ScrollAmount.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetHorizontalScrollPercent	Sets the horizontal scroll position as a percentage of the total content area within the pane.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetVerticalScrollPercent	Sets the vertical scroll position as a percentage of the total content area within the pane.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLPasswordBox Class

Description
Identifies a password box control.

Inheritance Hierarchy

- *SLBase*
  - SLPasswordBox

Syntax

```
'Declaration
Public Class SLPasswordBox _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsReadOnly	whether the password box is read-only
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all text from the password field.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetPosition	Sets the insertion point in the password field.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelection	Selects a range of text in the password box.
SetText	Replaces the text in the password field with the given text.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLPopup Class

Description
Identifies a popup control. Context menus are also treated as popup controls.

Inheritance Hierarchy

- *SLBase*
  - SLPopup

Syntax

```
'Declaration
Public Class SLPopup _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLProgressBar Class

Description
Identifies a progress bar control.

Inheritance Hierarchy

- *SLBase*
  - SLProgressBar

Syntax
```
'Declaration
Public Class SLProgressBar _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Maximum	the maximum progress value
Minimum	the minimum progress value
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLRadioButton Class

Description
Represents a button that allows a user to select a single option from a group of options.

Inheritance Hierarchy

- *SLBase*
  - SLRadioButton

Syntax
```
'Declaration
Public Class SLRadioButton _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsSelected	a value that indicates whether the radio button is selected.
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Deselects any selected items and then selects the current element.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLRepeatButton Class

Description
Identifies a repeat button control.

Inheritance Hierarchy

- *SLBase*
  - SLRepeatButton

Syntax

```
'Declaration
Public Class SLRepeatButton _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Clicks the button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLRichTextBox Class

Description
Identifies a rich text box control.

Inheritance Hierarchy

- *SLBase*
  - SLRichTextBox

Syntax
```
'Declaration
Public Class SLRichTextBox _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
SelectedText	the selected text
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Clear	Removes all text from the text box.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetPosition	Sets the insertion point in the text box.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelection	Selects a range of text in the text box.
SetText	Replaces the content in the text box with the given text.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a

Name	Description
	timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLSeparator Class

Description
Identifies a separator control.

Inheritance Hierarchy

- *SLBase*
  - SLSeparator

Syntax
```
'Declaration
Public Class SLSeparator _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)

Name	Description
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLSlider Class

Description
Identifies a progress bar control.

Inheritance Hierarchy

- *SLBase*

- SLSlider

Syntax

```
'Declaration
Public Class SLSlider _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
LargeChange	the value that is added to or subtracted from the Value property when a large change is made, such as with the PAGE DOWN key.
Maximum	the maximum progress value
Minimum	the minimum progress value
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
SmallChange	the value that is added to or subtracted from the Value property when a small change is made, such as with an arrow key.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DecreaseLarge	Decreases the value of the Slider by a large amount.
DecreaseSmall	Decreases the value of the Slider by a small amount.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
IncreaseLarge	Increases the value of the Slider by a large amount.
IncreaseSmall	Increases the value of the Slider by a small amount.

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetValue	Sets the value of the slider
SetValueToMaximum	Sets the Slider to the maximum position.
SetValueToMinimum	Sets the Slider to the minimum position.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLSpinner Class

Description

Identifies a spinner control.

A spinner control type consists of a set of buttons that enable a user to select from a set of items or set a numerical value from within a range.

Inheritance Hierarchy

- *SLBase*
  - SLSpinner

Syntax

```
'Declaration
Public Class SLSpinner _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)

Name	Description
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
NumericValue	the current numeric value of the spinner
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLSplitButton Class

Description
Identifies a split button control.

Inheritance Hierarchy

- *SLBase*

- SLSplitButton

Syntax

```
'Declaration
Public Class SLSplitButton _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLStatusBar Class

Description
Identifies a status bar control.

Inheritance Hierarchy

- *SLBase*
  - SLStatusBar

Syntax

```
'Declaration
Public Class SLStatusBar _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLTabControl Class

Description
Identifies a tab control.

Inheritance Hierarchy

- *SLBase*
  - SLTabControl

Syntax

```
'Declaration
Public Class SLTabControl _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Items	a complete list of tab items.
ItemTexts	a complete list of tab item texts.

Name	Description
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
SelectedIndex	the index of the selected item or negative one (-1) if the selection is empty.
SelectedItem	the selected item or `Nothing` if the selection is empty.
SelectedItemText	the text of the selected item or an empty string if the selection is empty.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item in the tab control.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLTabItem Class

Description
Identifies a tab item control.

Inheritance Hierarchy

- *SLBase*
  - SLTabItem

Syntax

```
'Declaration
Public Class SLTabItem _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsSelected	a value that indicates whether the tab item is selected
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects the tab item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLTable Class

Description
Identifies a table.

Inheritance Hierarchy

- *SLBase*
  - SLTable

Syntax

```
'Declaration
Public Class SLTable _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLTextBlock Class

Description
Identifies a control for displaying small amounts of flow content.

Inheritance Hierarchy

- *SLBase*
  - SLTextBlock

Syntax

```
'Declaration
Public Class SLTextBlock _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLTextBox Class

Description
Identifies a text box control.

Inheritance Hierarchy

- *SLBase*
  - SLTextBox

Syntax
```
'Declaration
Public Class SLTextBox _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsReadOnly	whether the text box is read-only
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all text from the text field.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetPosition	Sets the insertion point in the text field.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelection	Selects a range of text in the text box.
SetText	Replaces the text in the text field with the given text.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Examples

```
textBox.SetText("some text")
textBox.SetSelection(0, 4) ' selects 4 characters starting from the first one
textBox.TypeKeys("<Ctrl+C>") ' copies the selected text to the clipboard
Dim selectedText = Clipboard.GetText() ' gets the text from the clipboard
```

SLThumb Class

Description
Identifies a thumb control.

Inheritance Hierarchy

- *SLBase*
  - SLThumb

Syntax

```
'Declaration
Public Class SLThumb _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)

Name	Description
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLTitleBar Class

Description
Identifies the caption bar on a window.

Inheritance Hierarchy

- *SLBase*

- SLTitleBar

Syntax

```
'Declaration
Public Class SLTitleBar _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLToggleButton Class

Description
Identifies a toggle button control.

Inheritance Hierarchy

- *SLBase*
  - SLToggleButton

Syntax

```
'Declaration
Public Class SLToggleButton _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
ToggleState	the toggle state of the button
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Toggle	Cycles through the toggle states of the button in this order: On, Off and, if supported, Indeterminate.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLToolBar Class

Description
Identifies a toolbar control.

Inheritance Hierarchy

- *SLBase*
  - SLToolBar

Syntax

```
'Declaration
Public Class SLToolBar _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)

Name	Description
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLToolTip Class

Description
Identifies a tooltip control.

Inheritance Hierarchy

- *SLBase*
  - SLToolTip

Syntax

```
'Declaration
Public Class SLToolTip _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLTreeView Class

Description
Represents a control that displays hierarchical data in a tree structure that has items that can expand and collapse.

Inheritance Hierarchy

- *SLBase*
  - SLTreeView

Syntax

```
'Declaration
Public Class SLTreeView _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
HorizontallyScrollable	a value that indicates whether the tree view can scroll horizontally.
HorizontalScrollPercent	the current horizontal scroll position or negative one (-1) if there is no valid scroll position.
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
RootItems	a complete list of all root tree items.
RootItemTexts	a complete list of the texts of all root tree items.
SelectedItem	the selected item or returns `Nothing` if the selection is empty.
SelectedItemPath	the selected item path or an empty item path if nothing is selected
SelectedItemText	the text of the selected item or an empty string if the selection is empty.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticallyScrollable	a value that indicates whether the tree view can scroll vertically.

Name	Description
VerticalScrollPercent	the current vertical scroll position or negative one (-1) if there is no valid scroll position.
VisibleItemPaths	the visible items in the tree as item paths.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Collapse	Collapses the specified item.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Expand	Expands the specified item.
ExpandAll	Expands all items in the tree.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollHorizontal	Scrolls the currently visible region of the content area, horizontally, the specified ScrollAmount.
ScrollVertical	Scrolls the currently visible region of the content area, vertically, the specified ScrollAmount.
Select	Selects the specified item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetHorizontalScrollPercent	Sets the horizontal scroll position as a percentage of the total content area within the tree view.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetVerticalScrollPercent	Sets the vertical scroll position as a percentage of the total content area within the tree view.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLTreeViewItem Class

Description
Represents a selectable item in a tree control.

Inheritance Hierarchy

- *SLBase*
  - SLTreeViewItem

Syntax
```
'Declaration
Public Class SLTreeViewItem _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
IsExpanded	a value that indicates whether the nested items in a tree item are expanded or collapsed.
IsLeaf	a value that indicates whether the tree view item is a leaf item.
IsSelected	a value that indicates whether a tree item is selected.
Items	a complete list of items.
ItemTexts	a complete list of item texts.
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored

Name	Description
	in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Collapse	Collapses the item.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Expand	Expands the item.
ExpandSubTree	Expands the sub tree under this tree item.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)

Name	Description
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the tree item into view
Select	Selects the item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLValidationSummary Class

Description
Identifies a validation summary control.

Inheritance Hierarchy

- *SLBase*
  - SLValidationSummary

Syntax

```
'Declaration
Public Class SLValidationSummary _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLVerticalScrollBar Class

Description
Identifies a vertical scroll bar.

Inheritance Hierarchy

- *SLBase*
  - SLVerticalScrollBar

Syntax
```
'Declaration
Public Class SLVerticalScrollBar _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Maximum	the maximum scroll position
Minimum	the minimum scroll position
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)
ScrollPosition	the current scroll position
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Scroll	Scrolls the scroll bar by the specified amount
ScrollToMaximum	Scrolls the scroll bar to the bottom.
ScrollToMinimum	Scrolls the scroll bar to the top.
ScrollToPosition	Scrolls the scroll bar to the specific position
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SLWindow Class

Description
Identifies a window control.

Inheritance Hierarchy

- *SLBase*
  - SLWindow

Syntax

```
'Declaration
Public Class SLWindow _
Inherits SLBase
```

Properties

Name	Description
AutomationId	a string containing the UI Automation identifier (ID) for the element. (Inherited from *SLBase*)
ClassName	the simple class name of the element. (Inherited from *SLBase*)
IsEnabled	a value that indicates whether the element is enabled. (Inherited from *SLBase*)
Name	a string containing the UI Automation name for the element. (Inherited from *SLBase*)

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRawChildren	Gets the raw child element of this element. This method can be used to access elements that would otherwise not be recognized as children. Optionally one of the MSUIA control types can be specified to only return children of a certain type. (Inherited from *SLBase*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Java AWT and Swing Class Reference

Lists the available classes for testing Java AWT and Swing controls.

AbstractButton Class

Description
Defines common behaviors for buttons and menu items.

Inheritance Hierarchy

- *JComponent*
  - AbstractButton
    - *JButton*
    - *JMenuItem*
    - *JToggleButton*

Syntax

```
'Declaration
Public Class AbstractButton _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Applet Class

Description
Represents the Java Applet in the browser.

Inheritance Hierarchy
- *AWTContainer*
  - Applet

Syntax
```
'Declaration
Public Class Applet _
Inherits AWTContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AppletContainer Class

Description
Root element of an Applet in the browser.

Inheritance Hierarchy

- *AWTContainer*
  - AppletContainer

Syntax

```
'Declaration
Public Class AppletContainer _
Inherits AWTContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)

Name	Description
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTButton Class

Description
Represents a push button.

Inheritance Hierarchy

- *AWTComponent*
  - AWTButton

Syntax
```
'Declaration
Public Class AWTButton _
Inherits AWTComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTCanvas Class

Description

Represents an empty rectangular area of the screen where the application can draw or trap events from the user.

Inheritance Hierarchy

- *AWTComponent*
  - AWTCanvas

Syntax

```
'Declaration
Public Class AWTCanvas _
Inherits AWTComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTCheckbox Class

Description
Represents a check box.

Inheritance Hierarchy

- *AWTComponent*
  - AWTCheckbox

Syntax

```
'Declaration
Public Class AWTCheckbox _
Inherits AWTComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Checked	whether the check box is checked.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)

Name	Description
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Toggle	Toggles the check box state.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTCheckboxMenuItem Class

Description
Represents a check box control that can be inserted in a menu item.

Inheritance Hierarchy

- *AWTMenuItem*
  - AWTCheckboxMenuItem

Syntax
```
'Declaration
Public Class AWTCheckboxMenuItem _
Inherits AWTMenuItem
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTMenuItem*)
Checked	whether the check box of the menu item is checked.
Enabled	Whether the menu item is enabled. (Inherited from *AWTMenuItem*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box of the menu item.
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects the menu item. (Inherited from *AWTMenuItem*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Toggle	Toggles the check box state of the menu item.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box of the menu item.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTChoice Class

Description
Represents a combo box.

Inheritance Hierarchy

- *AWTComponent*
  - AWTChoice

Syntax

```
'Declaration
Public Class AWTChoice _
Inherits AWTComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)

Name	Description
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
ItemCount	the number of items in the combo box.
Items	the list of combo box items.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	the index of the selected item. -1 (4Test: 0) is returned, if no item is currently selected.
SelectedItem	the selected item. NULL is returned, if no item is currently selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItem	Returns the item at the specified index.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the specified item in the combo box.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any

Name	Description
	results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTComponent Class

Description
AWTComponent is the base class for Java AWT.

Inheritance Hierarchy

- *TestObject*
  - AWTComponent
    - *AWTButton*
    - *AWTCanvas*
    - *AWTCheckbox*
    - *AWTChoice*
    - *AWTContainer*
    - *AWTLabel*
    - *AWTList*
    - *AWTRadioButton*
    - *AWTScrollbar*
    - *AWTTextComponent*

Syntax
```
'Declaration
Public Class AWTComponent _
Inherits TestObject _
Implements IClickable, IFocusable, IKeyable, INativeWindow
```

Properties

Name	Description
AccessibleName	The accessible name of the component.
AccessibleRole	The accessible role of the component.
Background	The background color of the component.
Cursor	The name of the cursor set in the component.
Enabled	whether the component is enabled.

Name	Description
Focusable	whether the component can be focused.
Font	The font of the component.
Foreground	The foreground color of the component.
Height	The current height of the component.
Name	The name of the component.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane.

Name	Description
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTContainer Class

Description
Represents a component that can contain other AWT components.

Inheritance Hierarchy

- *AWTComponent*
  - AWTContainer
    - *Applet*
    - *AppletContainer*
    - *AWTScrollPane*
    - *AWTWindow*
    - *JComponent*
    - *OracleFormsContainer*
    - *SplitPaneDivider*

Syntax

```
'Declaration
Public Class AWTContainer _
Inherits AWTComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTDialog Class

Description
Represents a dialog box.

Inheritance Hierarchy

- *AWTWindow*
  - AWTDialog

- *JDialog*

Syntax

```
'Declaration
Public Class AWTDialog _
Inherits AWTWindow
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Modal	whether the dialog is modal.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Resizable	whether the dialog is resizable by the user.
Text	The text of the control. (Inherited from *TestObject*)
Title	the title of the dialog.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)

Name	Description
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTFrame Class

Description
Represents a top-level window with a title and a border.

Inheritance Hierarchy

- *AWTWindow*
  - AWTFrame
    - *JFrame*

Syntax

```
'Declaration
Public Class AWTFrame _
Inherits AWTWindow
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Resizable	whether the frame is resizable by the user.
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Title	the title of the frame.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)

Name	Description
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTHorizontalScrollbar Class

Description
Represents a horizontal scroll bar.

Inheritance Hierarchy
- *AWTScrollbar*
  - AWTHorizontalScrollbar

Syntax
```
'Declaration
Public Class AWTHorizontalScrollbar _
Inherits AWTScrollbar
```

Properties
Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)

Name	Description
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	the block increment value of the scroll bar. (Inherited from *IAWTScroller*)
Position	the current position of the scroll bar. (Inherited from *IAWTScroller*)
Range	the range value of the scroll bar. (Inherited from *IAWTScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IAWTScroller*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
Page	Increments the current scroll position by specified number of pages. (Inherited from *IAWTScroller*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IAWTScroller*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Step	Increments the current scroll position by specified steps. (Inherited from *IAWTScroller*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTLabel Class

Description
Represents a class for placing text in a container.

Inheritance Hierarchy

- *AWTComponent*
  - AWTLabel

Syntax

```
'Declaration
Public Class AWTLabel _
Inherits AWTComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Alignment	the alignment of the label.
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)

Name	Description
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTList Class

Description
Represents a list box.

Inheritance Hierarchy

- *AWTComponent*
  - AWTList

Syntax

```
'Declaration
Public Class AWTList _
Inherits AWTComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
ItemCount	the number of list items.
Items	a list of items in the list.
MultipleMode	whether the list allows multiple selection.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	the index of the currently selected item. -1 (4Test: 0) is returned, if no item is currently selected.
SelectedIndices	the indices of the currently selected items. An empty list is returned, if no items are currently selected.

Name	Description
SelectedItem	the currently selected list item. NULL is returned, if no item is currently selected.
SelectedItems	the currently selected list items. An empty list is returned, if no items are currently selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Deselect	Deselects the specified list item.
DeselectAll	Deselects all list items.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleClickItem	Double-clicks a list item.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItem	Returns the list item at the specified index.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the specified item.
SelectAll	Selects all items in the list.
SelectRange	Selects all items within the specified range.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the

Name	Description
	timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTMenu Class

Description
Represents a menu component that is accessed from a menu bar.

Inheritance Hierarchy

- *AWTMenuItem*
  - AWTMenu

Syntax
```
'Declaration
Public Class AWTMenu _
Inherits AWTMenuItem
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTMenuItem*)
Enabled	Whether the menu item is enabled. (Inherited from *AWTMenuItem*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects the menu item. (Inherited from *AWTMenuItem*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object,

Name	Description
	for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTMenuComponent Class

Description
Represents the base class of all menu-related controls.

Inheritance Hierarchy

- *TestObject*
  - AWTMenuComponent
    - *AWTMenuItem*

Syntax
```
'Declaration
Public Class AWTMenuComponent _
Inherits TestObject _
Implements IClickable, IFocusable, IKeyable
```

Properties

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown.

Name	Description
	The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTMenuItem Class

Description
Represents a menu item.

Inheritance Hierarchy

- *AWTMenuComponent*
  - AWTMenuItem
    - *AWTCheckboxMenuItem*
    - *AWTMenu*

Syntax
```
'Declaration
Public Class AWTMenuItem _
Inherits AWTMenuComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component.
Enabled	Whether the menu item is enabled.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects the menu item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTRadioButton Class

Description
Represents a radio button.

Inheritance Hierarchy

- *AWTComponent*
  - AWTRadioButton

Syntax

```
'Declaration
Public Class AWTRadioButton _
Inherits AWTComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)

Name	Description
Background	The background color of the component. (Inherited from *AWTComponent*)
Checked	Whether the radio button is checked.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the radio button.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTScrollbar Class

Description

Represents a scroll bar. This class includes scroll bars that are parts of controls, such as the scroll bar on a list box.

Inheritance Hierarchy

- *AWTComponent*
  - AWTScrollbar
    - *AWTHorizontalScrollbar*
    - *AWTVerticalScrollbar*

Syntax

```
'Declaration
Public Class AWTScrollbar _
Inherits AWTComponent _
Implements IAWTScroller
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)

Name	Description
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	the block increment value of the scroll bar. (Inherited from *IAWTScroller*)
Position	the current position of the scroll bar. (Inherited from *IAWTScroller*)
Range	the range value of the scroll bar. (Inherited from *IAWTScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IAWTScroller*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
Page	Increments the current scroll position by specified number of pages. (Inherited from *IAWTScroller*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IAWTScroller*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Step	Increments the current scroll position by specified steps. (Inherited from *IAWTScroller*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any

Name	Description
	results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTScrollPane Class

Description
Represents a class that implements automatic horizontal or vertical scrolling for a single child component.

Inheritance Hierarchy

- *AWTContainer*
  - AWTScrollPane

Syntax

```
'Declaration
Public Class AWTScrollPane _
Inherits AWTContainer _
Implements IAWTScrollable
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)

Name	Description
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	the current scroll position of the scrollable component. (Inherited from *IAWTScrollable*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Gets the horizontal scroll bar. (Inherited from *IAWTScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Gets the vertical scroll bar. (Inherited from *IAWTScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IAWTScrollable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the

Name	Description
	timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTTextArea Class

Description
Represents a text area.

Inheritance Hierarchy

- *AWTTextComponent*
  - AWTTextArea

Syntax

```
'Declaration
Public Class AWTTextArea _
Inherits AWTTextComponent _
Implements IAWTScrollable
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text component. (Inherited from *AWTTextComponent*)
Columns	the number of columns in the text area.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Editable	whether the text of the text component can be changed by the user. (Inherited from *AWTTextComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)

Name	Description
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	the current scroll position of the scrollable component. (Inherited from *IAWTScrollable*)
Rows	The number of rows in the text area.
SelectedText	the current selected text. (Inherited from *AWTTextComponent*)
SelectionEnd	the end position of the selected text. (Inherited from *AWTTextComponent*)
SelectionStart	the start position of the selected text. (Inherited from *AWTTextComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from the text component. (Inherited from *AWTTextComponent*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Gets the horizontal scroll bar. (Inherited from *IAWTScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Gets the vertical scroll bar. (Inherited from *IAWTScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Insert	Inserts the specified text at the specified position.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReplaceRange	Replaces text between the specified start and end positions with the specified replacement text.
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IAWTScrollable*)
SelectAll	Selects all the text in the text component. (Inherited from *AWTTextComponent*)
SetCaretPosition	Sets the position of the text insertion caret for this text component. (Inherited from *AWTTextComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range. (Inherited from *AWTTextComponent*)
SetText	Sets the text of the text component to the specified text. (Inherited from *AWTTextComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTTextComponent Class

Description
Represents the base class for the AWTTextField and AWTTextArea classes.

Inheritance Hierarchy

- *AWTComponent*
  - AWTTextComponent
    - *AWTTextArea*
    - *AWTTextField*

Syntax

```
'Declaration
Public Class AWTTextComponent _
Inherits AWTComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)

Name	Description
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text component.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Editable	whether the text of the text component can be changed by the user.
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedText	the current selected text.
SelectionEnd	the end position of the selected text.
SelectionStart	the start position of the selected text.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from the text component.
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SelectAll	Selects all the text in the text component.
SetCaretPosition	Sets the position of the text insertion caret for this text component.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range.
SetText	Sets the text of the text component to the specified text.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by

Name	Description
	the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTTextField Class

Description
Represents single-line and multi-line text field controls whose text can be modified by the user.

Inheritance Hierarchy

- *AWTTextComponent*
  - AWTTextField

Syntax
```
'Declaration
Public Class AWTTextField _
Inherits AWTTextComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)

Name	Description
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text component. (Inherited from *AWTTextComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
EchoChar	the character that is displayed instead of the characters that are actually typed.
EchoCharSet	whether *EchoChar* is set to a value other than *0*.
Editable	whether the text of the text component can be changed by the user. (Inherited from *AWTTextComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
IsPassword	whether the text field is a password text field.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedText	the current selected text. (Inherited from *AWTTextComponent*)
SelectionEnd	the end position of the selected text. (Inherited from *AWTTextComponent*)
SelectionStart	the start position of the selected text. (Inherited from *AWTTextComponent*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha

Name	Description
	channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from the text component. (Inherited from *AWTTextComponent*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SelectAll	Selects all the text in the text component. (Inherited from *AWTTextComponent*)
SetCaretPosition	Sets the position of the text insertion caret for this text component. (Inherited from *AWTTextComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range. (Inherited from *AWTTextComponent*)
SetText	Sets the text of the text component to the specified text. (Inherited from *AWTTextComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTVerticalScrollbar Class

Description
Represents a vertical scroll bar.

Inheritance Hierarchy

- *AWTScrollbar*
  - AWTVerticalScrollbar

Syntax

```
'Declaration
Public Class AWTVerticalScrollbar _
Inherits AWTScrollbar
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	the block increment value of the scroll bar. (Inherited from *IAWTScroller*)
Position	the current position of the scroll bar. (Inherited from *IAWTScroller*)
Range	the range value of the scroll bar. (Inherited from *IAWTScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IAWTScroller*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
Page	Increments the current scroll position by specified number of pages. (Inherited from *IAWTScroller*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IAWTScroller*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Step	Increments the current scroll position by specified steps. (Inherited from *IAWTScroller*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

AWTWindow Class

Description
Represents a top-level window with no borders and no menubar.

Inheritance Hierarchy

- *AWTContainer*
  - AWTWindow
    - *AWTDialog*
    - *AWTFrame*
    - *JWindow*

Syntax

```
'Declaration
Public Class AWTWindow _
```

```
Inherits AWTContainer _
Implements IMoveable
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

BasicArrowButton Class

Description
Represents a push button with an arrow in one of the cardinal directions.

Inheritance Hierarchy

- *JButton*

- BasicArrowButton

Syntax

```
'Declaration
Public Class BasicArrowButton _
Inherits JButton
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Direction	the direction the arrow points to.
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button. (Inherited from *AbstractButton*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by

Name	Description
	the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

IAWTScrollable Interface

Description
Represents an interface for AWT objects that can have two scroll bars.

Inheritance Hierarchy

- *IBaseScrollable*
  - IAWTScrollable

Syntax

```
'Declaration
Public Class IAWTScrollable _
Inherits IBaseScrollable
```

Properties

Name	Description
Position	the current scroll position of the scrollable component. (Inherited from *IBaseScrollable*)

Methods

Name	Description
GetHorizontalBar	Gets the horizontal scroll bar.
GetVerticalBar	Gets the vertical scroll bar.
ScrollTo	Scrolls to the specified position. (Inherited from *IBaseScrollable*)

IAWTScroller Interface

Description
Represents an interface for scrollable AWT objects.

Syntax
```
'Declaration
Public Class IAWTScroller
```

Properties

Name	Description
PageSize	the block increment value of the scroll bar.
Position	the current position of the scroll bar.
Range	the range value of the scroll bar.
StepSize	the unit increment value of the scroll bar.

Methods

Name	Description
Page	Increments the current scroll position by specified number of pages.
ScrollTo	Scrolls to the specified position.
Step	Increments the current scroll position by specified steps.

IBaseScrollable Interface

Description
Represents an interface for other scrollable controls to avoid duplication.

Syntax
```
'Declaration
Public Class IBaseScrollable
```

Properties

Name	Description
Position	the current scroll position of the scrollable component.

Methods

Name	Description
ScrollTo	Scrolls to the specified position.

IOracleFormsMenuBase Interface

Description
Represents a control that defines choices for users to select.

Syntax
```
'Declaration
Public Class IOracleFormsMenuBase
```

Properties

Name	Description
Checked	Whether the MenuItem is checked.

IOracleFormsScrollable Interface

Description
Represents an interface for scrollable oracle forms objects.

Syntax
```
'Declaration
Public Class IOracleFormsScrollable
```

Properties

Name	Description
HorizontalScrollbarPageSize	the block increment value of the scroll bar.
HorizontalScrollbarPosition	the current position of the scroll bar.
HorizontalScrollbarRange	the range value of the scroll bar.
HorizontalScrollbarStepSize	the unit increment value of the scroll bar.
VerticalScrollbarPageSize	the block increment value of the scroll bar.
VerticalScrollbarPosition	the current position of the scroll bar.
VerticalScrollbarRange	the range value of the scroll bar.
VerticalScrollbarStepSize	the unit increment value of the scroll bar.

Methods

Name	Description
GetHorizontalScrollbar	Gets the horizontal scroll bar.
GetVerticalScrollbar	Gets the vertical scroll bar.

Name	Description
HorizontalScrollbarPage	Increments the current scroll position by specified number of pages.
HorizontalScrollbarScrollTo	Scrolls to the specified position.
HorizontalScrollbarScrollToMax	Scrolls to the maximum position of the scroller.
HorizontalScrollbarScrollToMin	Scrolls to the minimum position of the scroller.
HorizontalScrollbarStep	Increments the current scroll position by specified steps.
VerticalScrollbarPage	Increments the current scroll position by specified number of pages.
VerticalScrollbarScrollTo	Scrolls to the specified position.
VerticalScrollbarScrollToMax	Scrolls to the maximum position of the scroller.
VerticalScrollbarScrollToMin	Scrolls to the minimum position of the scroller.
VerticalScrollbarStep	Increments the current scroll position by specified steps.

IOracleFormsScroller Interface

Description
Represents an interface for scrollable oracle forms objects.

Inheritance Hierarchy

- *IAWTScroller*
  - IOracleFormsScroller

Syntax
```
'Declaration
Public Class IOracleFormsScroller _
Inherits IAWTScroller
```

Properties

Name	Description
PageSize	the block increment value of the scroll bar. (Inherited from *IAWTScroller*)
Position	the current position of the scroll bar. (Inherited from *IAWTScroller*)
Range	the range value of the scroll bar. (Inherited from *IAWTScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IAWTScroller*)

Methods

Name	Description
Page	Increments the current scroll position by specified number of pages. (Inherited from *IAWTScroller*)
ScrollTo	Scrolls to the specified position. (Inherited from *IAWTScroller*)

Name	Description
ScrollToMax	Scrolls to the maximum position of the scroller.
ScrollToMin	Scrolls to the minimum position of the scroller.
Step	Increments the current scroll position by specified steps. (Inherited from *IAWTScroller*)

JButton Class

Description
Represents a push button.

Inheritance Hierarchy

- *AbstractButton*
  - JButton
    - *BasicArrowButton*

Syntax

```
'Declaration
Public Class JButton _
Inherits AbstractButton
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button. (Inherited from *AbstractButton*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JCheckBox Class

Description
Represents a check box.

Inheritance Hierarchy

- *JToggleButton*
  - JCheckBox

Syntax

```
'Declaration
Public Class JCheckBox _
Inherits JToggleButton
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Selected	whether the toggle button is selected. (Inherited from *JToggleButton*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button. (Inherited from *AbstractButton*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Toggle	Toggles the button. (Inherited from *JToggleButton*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JCheckBoxMenuItem Class

Description
Represents a check box control that can be included in a menu item.

Inheritance Hierarchy

- *JMenuItem*
  - JCheckBoxMenuItem

Syntax

```
'Declaration
Public Class JCheckBoxMenuItem _
Inherits JMenuItem
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Checked	whether the check box in a menu item is checked.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button. (Inherited from *AbstractButton*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JColorChooser Class

Description
Represents a color chooser.

Inheritance Hierarchy

- *JComponent*
  - JColorChooser

Syntax

```
'Declaration
Public Class JColorChooser _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Color	the current color value from the color chooser.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)

Name	Description
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetColor	Sets the current color of the color chooser to the specified color.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the

Name	Description
	timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JComboBox Class

Description
Represents a combo box.

Inheritance Hierarchy

- *JComponent*
  - JComboBox

Syntax

```
'Declaration
Public Class JComboBox _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Editable	whether the combo box is editable.
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
ItemCount	the number of items in the list.
Items	the list of all contained items.
Name	The name of the component. (Inherited from *AWTComponent*)

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	the index of the currently selected item.
SelectedItem	the current selected item.
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItem	Returns the item at the specified index.

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the specified item.

Name	Description
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JComponent Class

Description
Represents a class for all components except top-level containers.

Inheritance Hierarchy

- *AWTContainer*
  - JComponent
    - *AbstractButton*
    - *JColorChooser*
    - *JComboBox*
    - *JLabel*
    - *JLayeredPane*
    - *JList*
    - *JMenuBar*
    - *JPanel*
    - *JPopupMenu*
    - *JProgressBar*
    - *JRootPane*
    - *JScrollBar*
    - *JScrollPane*
    - *JSlider*
    - *JSpinner*
    - *JSplitPane*
    - *JTabbedPane*
    - *JTable*
    - *JTableHeader*
    - *JTextComponent*
    - *JToolBar*
    - *JTree*
    - *JViewport*

Syntax

```
'Declaration
Public Class JComponent _
Inherits AWTContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)

Name	Description
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component.
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JDesktopPane Class

Description
Represents a desktop pane.

Inheritance Hierarchy

- *JLayeredPane*
  - JDesktopPane

Syntax

```
'Declaration
Public Class JDesktopPane _
Inherits JLayeredPane
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JDialog Class

Description
Represents a dialog box.

Inheritance Hierarchy

- *AWTDialog*
  - JDialog

Syntax

```
'Declaration
Public Class JDialog _
Inherits AWTDialog
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Modal	whether the dialog is modal. (Inherited from *AWTDialog*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Resizable	whether the dialog is resizable by the user. (Inherited from *AWTDialog*)
Text	The text of the control. (Inherited from *TestObject*)
Title	the title of the dialog. (Inherited from *AWTDialog*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)

Name	Description
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JEditorPane Class

Description
Represents a text component where you can edit different types of content.

Inheritance Hierarchy

- *JTextComponent*
  - JEditorPane
    - *JTextPane*

Syntax

```
'Declaration
Public Class JEditorPane _
Inherits JTextComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text component. (Inherited from *JTextComponent*)
ContentType	the type of content that this editor is currently set to deal with.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Editable	whether the text of the text component can be changed by the user. (Inherited from *JTextComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
FormattedText	the text of the editor pane.

Name	Description
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Page	the current URL being displayed.
SelectedText	the current selected text. (Inherited from *JTextComponent*)
SelectionEnd	the end position of the selection. (Inherited from *JTextComponent*)
SelectionStart	the start position of the selection. (Inherited from *JTextComponent*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from text component. (Inherited from *JTextComponent*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SelectAll	Selects all the content in the text component. (Inherited from *JTextComponent*)
SetCaretPosition	Sets the position of the text insertion caret for this text component. (Inherited from *JTextComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range. (Inherited from *JTextComponent*)
SetText	Sets the text content to the specified text. (Inherited from *JTextComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JFrame Class

Description
Represents a class for an extended version of AWTFrame that adds support for Java Swing.

Inheritance Hierarchy

- *AWTFrame*
  - JFrame

Syntax

```
'Declaration
Public Class JFrame _
Inherits AWTFrame
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)

Name	Description
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Resizable	whether the frame is resizable by the user. (Inherited from *AWTFrame*)
Text	The text of the control. (Inherited from *TestObject*)
Title	the title of the frame. (Inherited from *AWTFrame*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)

Name	Description
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JHorizontalScrollBar Class

Description
Represents a horizontal scroll bar.

Inheritance Hierarchy

- *JScrollBar*
  - JHorizontalScrollBar

Syntax
```
'Declaration
Public Class JHorizontalScrollBar _
Inherits JScrollBar
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)

Name	Description
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	the block increment value of the scroll bar. (Inherited from *IAWTScroller*)
Position	the current position of the scroll bar. (Inherited from *IAWTScroller*)
Range	the range value of the scroll bar. (Inherited from *IAWTScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IAWTScroller*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
Page	Increments the current scroll position by specified number of pages. (Inherited from *IAWTScroller*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IAWTScroller*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Step	Increments the current scroll position by specified steps. (Inherited from *IAWTScroller*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by

Name	Description
	the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JLabel Class

Description
Represents a display area for displaying a short text, image, or both.

Inheritance Hierarchy

- *JComponent*
  - JLabel

Syntax
```
'Declaration
Public Class JLabel _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)

Name	Description
Alignment	the alignment of the label.
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JLayeredPane Class

Description
Represents a layered pane.

Inheritance Hierarchy

- *JComponent*
  - JLayeredPane
    - *JDesktopPane*

Syntax

```
'Declaration
Public Class JLayeredPane _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)

Name	Description
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JList Class

Description
Represents a list box.

Inheritance Hierarchy

- *JComponent*
  - JList

Syntax

```
'Declaration
Public Class JList _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
ItemCount	the number of list items.
Items	the list of all contained list items.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	the index of the selected list item.
SelectedIndices	the list of indices of the selected list items.
SelectedItem	the name or index of the selected list item.
SelectedItems	the list of names or indices of the selected list items.
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
ClickItem	Clicks the specified list item.
Deselect	Deselects the specified list item.
DeselectAll	Deselects all list items.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleClickItem	Double-clicks a list item.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
FindItem	Returns the index of an item in the list.
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItem	Returns the item at the specified index.
GetItemRect	Returns the size and position of the item relative to the list.
GetItemValue	Returns the value object of the list item.

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollItemIntoView	Scrolls an item into view.

Name	Description
Select	Selects a list item.
SelectAll	Selects all list elements.
SelectRange	Selects a specified range of list elements.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JMenu Class

Description
Represents a menu. A menu is a menu item that has child items.

Inheritance Hierarchy

- *JMenuItem*
  - JMenu

Syntax

```
'Declaration
Public Class JMenu _
Inherits JMenuItem
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button. (Inherited from *AbstractButton*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JMenuBar Class

Description
Represents a menu bar.

Inheritance Hierarchy

- *JComponent*
  - JMenuBar

Syntax

```
'Declaration
Public Class JMenuBar _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JMenuItem Class

Description
Represents a menu item.

Inheritance Hierarchy

- *AbstractButton*
  - JMenuItem
    - *JCheckBoxMenuItem*
    - *JMenu*
    - *JRadioButtonMenuItem*

Syntax

```
'Declaration
Public Class JMenuItem _
Inherits AbstractButton
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)

Name	Description
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button. (Inherited from *AbstractButton*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JPanel Class

Description
Represents a container class for lightweight components.

Inheritance Hierarchy

- *JComponent*
  - JPanel

Syntax

```
'Declaration
Public Class JPanel _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JPasswordField Class

Description
Represents a single-line field that does not show the original characters.

Inheritance Hierarchy

- *JTextField*
  - JPasswordField

Syntax

```
'Declaration
Public Class JPasswordField _
Inherits JTextField
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Alignment	the alignment of the text field. (Inherited from *JTextField*)
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text component. (Inherited from *JTextComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
EchoChar	the character that is displayed instead of the characters that are actually typed.
EchoCharSet	whether *EchoChar* is set to a value other than *0*.
Editable	whether the text of the text component can be changed by the user. (Inherited from *JTextComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedText	the current selected text. (Inherited from *JTextComponent*)
SelectionEnd	the end position of the selection. (Inherited from *JTextComponent*)
SelectionStart	the start position of the selection. (Inherited from *JTextComponent*)

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from text component. (Inherited from *JTextComponent*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SelectAll	Selects all the content in the text component. (Inherited from *JTextComponent*)
SetCaretPosition	Sets the position of the text insertion caret for this text component. (Inherited from *JTextComponent*)

Name	Description
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range. (Inherited from *JTextComponent*)
SetText	Sets the text content to the specified text. (Inherited from *JTextComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JPopupMenu Class

Description
Represents a context menu.

Inheritance Hierarchy

- *JComponent*
  - JPopupMenu

Syntax

```
'Declaration
Public Class JPopupMenu _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JProgressBar Class

Description
Displays the progress of an operation.

Inheritance Hierarchy

- *JComponent*
  - JProgressBar

Syntax

```
'Declaration
Public Class JProgressBar _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Indeterminate	whether the progress bar is indeterminate.
Maximum	the progress bar's maximum value.
Minimum	the progress bar's minimum value.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
ProgressString	the string representation of the current progress.
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JRadioButton Class

Description
Represents a radio button.

Inheritance Hierarchy

- *JToggleButton*
  - JRadioButton

Syntax

```
'Declaration
Public Class JRadioButton _
Inherits JToggleButton
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)

Name	Description
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Selected	whether the toggle button is selected. (Inherited from *JToggleButton*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the radio button.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button. (Inherited from *AbstractButton*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Toggle	Toggles the button. (Inherited from *JToggleButton*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use

Name	Description
	WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JRadioButtonMenuItem Class

Description
Represents a radio button menu item.

Inheritance Hierarchy

- *JMenuItem*
  - JRadioButtonMenuItem

Syntax

```
'Declaration
Public Class JRadioButtonMenuItem _
Inherits JMenuItem
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Checked	whether the radio button in a menu item is checked.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)

Name	Description
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button. (Inherited from *AbstractButton*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JRootPane Class

Description
Represents a root pane.

Inheritance Hierarchy

- *JComponent*
  - JRootPane

Syntax

```
'Declaration
Public Class JRootPane _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JScrollBar Class

Description
Represents a scroll bar.

Inheritance Hierarchy

- *JComponent*
  - JScrollBar
    - *JHorizontalScrollBar*
    - *JVerticalScrollBar*

Syntax

```
'Declaration
Public Class JScrollBar _
```

```
Inherits JComponent _
Implements IAWTScroller
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	the block increment value of the scroll bar. (Inherited from *IAWTScroller*)
Position	the current position of the scroll bar. (Inherited from *IAWTScroller*)
Range	the range value of the scroll bar. (Inherited from *IAWTScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IAWTScroller*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
Page	Increments the current scroll position by specified number of pages. (Inherited from *IAWTScroller*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IAWTScroller*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Step	Increments the current scroll position by specified steps. (Inherited from *IAWTScroller*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JScrollPane Class

Description
Represents a scroll pane which allows scrolling its content which may be larger than the pane itself.

Inheritance Hierarchy

- *JComponent*
  - JScrollPane

Syntax

```
'Declaration
Public Class JScrollPane _
Inherits JComponent _
Implements IBaseScrollable
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	the current scroll position of the scrollable component. (Inherited from *IBaseScrollable*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Gets the horizontal scroll bar.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Gets the vertical scroll bar.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IBaseScrollable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JSlider Class

Description
Represents a slider control which can select a value within a range.

Inheritance Hierarchy

- *JComponent*
  - JSlider

Syntax

```
'Declaration
Public Class JSlider _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)

Name	Description
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Inverted	the value is *true* if the value-range shown for the slider is reversed.
Maximum	the maximum value supported by the slider.
Minimum	the minimum value supported by the slider.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	the position of the slider.
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetPosition	Sets the current position of the slider.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JSpinner Class

Description
Represents a spinner control which can iterate a sequence of values.

Inheritance Hierarchy

- *JComponent*
  - JSpinner

Syntax

```
'Declaration
Public Class JSpinner _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)

Name	Description
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Next	Sets the spinner to the object in the sequence that comes after the object returned by the Value property.
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Previous	Sets the spinner to the object in the sequence that comes before the object returned by the Value property.

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the

Name	Description
	timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JSplitPane Class

Description
Represents a split pane control which divides two components horizontally or vertically.

Inheritance Hierarchy

- *JComponent*
  - JSplitPane

Syntax
```
'Declaration
Public Class JSplitPane _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
DividerLocation	the current divider location.
DividerSize	the size of the divider.
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
MaximumDividerLocation	the maximum location of the divider.
MinimumDividerLocation	the minimum location of the divider.
Name	The name of the component. (Inherited from *AWTComponent*)

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetPosition	Sets the position of the divider.

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JTabbedPane Class

Description
Represents a tabbed pane.

Inheritance Hierarchy

- *JComponent*
  - JTabbedPane

Syntax
```
'Declaration
Public Class JTabbedPane _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedTab	the currently selected tab.
SelectedTabIndex	the index of the currently selected tab or -1 (4Test: 0) if no tab is selected.
TabCount	the number of tabs.
TabRunCount	the number of tab runs currently used to display the tabs.
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetTabIndex	Returns the tab index.
GetTabTitle	Returns the tab title.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsTabEnabled	Returns whether or not the tab is currently enabled.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the specified tab.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JTable Class

Description
Represents a table control which has rows and columns.

Inheritance Hierarchy

- *JComponent*
  - JTable

Syntax
```
'Declaration
Public Class JTable _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
AllCellTexts	the text values of all cells in the table. The cells are grouped in rows, where each row contains the texts of its cells.
AllCellValues	the values of all cells in the table. The cells are grouped in rows, where each row contains the values of its cells.
Background	The background color of the component. (Inherited from *AWTComponent*)
ColumnCount	the number of columns in the table.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
EditingColumn	the name or index of the column that contains the cell currently being edited.
EditingRow	the index of the row that contains the cell currently being edited.
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
RowCount	the number of rows that can be shown in the table.
SelectedColumn	the index or name of the selected column. -1 (4Test: 0) is returned if no column is selected.
SelectedColumns	the list of indices of all selected columns.
SelectedRow	the index of the selected row, -1 (4Test: 0) is returned if no row is selected.
SelectedRows	the list of indices of all selected rows.
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearSelection	Deselects all selected columns and rows.
Click	Clicks on the object. (Inherited from *IClickable*)
ClickCell	Clicks the specified cell in the table.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleClickCell	Double-clicks the specified cell in the table.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
FindCell	Returns the cell containing the specified text.
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCellText	Returns the cell text.
GetCellValue	Returns the cell value.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetColumnName	Returns the name of the column at the specified index.
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsCellEditable	Returns true if the specified cell is editable.
IsCellSelected	Returns true if the specified cell is selected; otherwise false.
IsColumnSelected	Returns true if the specified column is selected.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
IsRowSelected	Returns true if the specified row is selected.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MoveColumn	Moves the column to the position of the target column.
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ResizeColumn	Resizes the specified column.
ScrollCellIntoView	Scrolls the specified cell into view.
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SelectAll	Selects all rows, columns, and cells in the table.
SelectColumn	Selects the specified column.
SelectRow	Selects the specified row.
SetCellValue	Sets the specified value for the specified cell in the table.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JTableHeader Class

Description
Represents table header class for the table component.

Inheritance Hierarchy

- *JComponent*
  - JTableHeader

Syntax
```
'Declaration
Public Class JTableHeader _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)

Name	Description
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
ClickColumnHeader	Clicks the column header of the table.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object,

Name	Description
	for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JTextArea Class

Description
Represents a multiple-line area control that displays plain text.

Inheritance Hierarchy

- *JTextComponent*
  - JTextArea

Syntax

```
'Declaration
Public Class JTextArea _
Inherits JTextComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text component. (Inherited from *JTextComponent*)
Columns	the number of columns in the text area.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Editable	whether the text of the text component can be changed by the user. (Inherited from *JTextComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)

Name	Description
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Rows	The number of rows in the text area.
SelectedText	the current selected text. (Inherited from *JTextComponent*)
SelectionEnd	the end position of the selection. (Inherited from *JTextComponent*)
SelectionStart	the start position of the selection. (Inherited from *JTextComponent*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from text component. (Inherited from *JTextComponent*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Insert	Inserts the specified text at the specified position.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReplaceRange	Replaces text within the specified range with the new text specified.
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SelectAll	Selects all the content in the text component. (Inherited from *JTextComponent*)
SetCaretPosition	Sets the position of the text insertion caret for this text component. (Inherited from *JTextComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range. (Inherited from *JTextComponent*)
SetText	Sets the text content to the specified text. (Inherited from *JTextComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a

Name	Description
	timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JTextComponent Class

Description

Represents a text control that can be marked with attributes that are represented graphically.

Inheritance Hierarchy

- *JComponent*
  - JTextComponent
    - *JEditorPane*
    - *JTextArea*
    - *JTextField*

Syntax

```
'Declaration
Public Class JTextComponent _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text component.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Editable	whether the text of the text component can be changed by the user.
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedText	the current selected text.
SelectionEnd	the end position of the selection.
SelectionStart	the start position of the selection.
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from text component.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SelectAll	Selects all the content in the text component.
SetCaretPosition	Sets the position of the text insertion caret for this text component.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range.
SetText	Sets the text content to the specified text.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JTextField Class

Description

Represents a single-line text field control whose text can be modified by the user.

Inheritance Hierarchy

- *JTextComponent*
  - JTextField
    - *JPasswordField*

Syntax

```
'Declaration
Public Class JTextField _
Inherits JTextComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Alignment	the alignment of the text field.
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text component. (Inherited from *JTextComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Editable	whether the text of the text component can be changed by the user. (Inherited from *JTextComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedText	the current selected text. (Inherited from *JTextComponent*)
SelectionEnd	the end position of the selection. (Inherited from *JTextComponent*)
SelectionStart	the start position of the selection. (Inherited from *JTextComponent*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location

Name	Description
	on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from text component. (Inherited from *JTextComponent*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SelectAll	Selects all the content in the text component. (Inherited from *JTextComponent*)
SetCaretPosition	Sets the position of the text insertion caret for this text component. (Inherited from *JTextComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range. (Inherited from *JTextComponent*)
SetText	Sets the text content to the specified text. (Inherited from *JTextComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JTextPane Class

Description
Represents a text pane control that can be marked with attributes that are represented graphically.

Inheritance Hierarchy

- *JEditorPane*
  - JTextPane

Syntax

```
'Declaration
Public Class JTextPane _
Inherits JEditorPane
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text component. (Inherited from *JTextComponent*)
CharacterAttributes	the character attributes in effect at the location of the caret.
ContentType	the type of content that this editor is currently set to deal with. (Inherited from *JEditorPane*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Editable	whether the text of the text component can be changed by the user. (Inherited from *JTextComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
FormattedText	the text of the editor pane. (Inherited from *JEditorPane*)
Height	The current height of the component. (Inherited from *AWTComponent*)
LogicalStyle	the logical style assigned to the current paragraph in effect at the location of the caret.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Page	the current URL being displayed. (Inherited from *JEditorPane*)
ParagraphAttributes	the attributes of the current paragraph in effect at the location of the caret.
SelectedText	the current selected text. (Inherited from *JTextComponent*)
SelectionEnd	the end position of the selection. (Inherited from *JTextComponent*)

Name	Description
SelectionStart	the start position of the selection. (Inherited from *JTextComponent*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from text component. (Inherited from *JTextComponent*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SelectAll	Selects all the content in the text component. (Inherited from *JTextComponent*)

Name	Description
SetCaretPosition	Sets the position of the text insertion caret for this text component. (Inherited from *JTextComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range. (Inherited from *JTextComponent*)
SetText	Sets the text content to the specified text. (Inherited from *JTextComponent*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JToggleButton Class

Description
Represents a two-state button.

Inheritance Hierarchy

- *AbstractButton*
  - JToggleButton
    - *JCheckBox*
    - *JRadioButton*

Syntax

```
'Declaration
Public Class JToggleButton _
Inherits AbstractButton
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Selected	whether the toggle button is selected.
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button. (Inherited from *AbstractButton*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Toggle	Toggles the button.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JToolBar Class

Description
Represents a toolbar.

Inheritance Hierarchy

- *JComponent*
  - JToolBar

Syntax

```
'Declaration
Public Class JToolBar _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JTree Class

Description
Represents a tree.

Inheritance Hierarchy

- *JComponent*
  - JTree

Syntax
```
'Declaration
Public Class JTree _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)

Name	Description
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
IsEditable	whether the tree is editable.
IsEditing	whether the tree is being edited.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedItemPath	the selected item path.
SelectedItemPaths	the list of selected item paths.
SelectedItemText	the selected item text.
SelectionCount	the number of tree nodes selected.
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VisibleItemPaths	the visible items in the tree as item paths.
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CancelEditing	Cancels the current editing session.
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearSelection	Clears the selection.
Click	Clicks on the object. (Inherited from *IClickable*)
ClickNode	Clicks on a tree node specified by the item path.
Collapse	Collapses the tree node identified by the specified item path.
CollapseAll	Collapses all tree nodes.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleClickNode	Double-clicks on a tree node specified by the item path.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Expand	Expands the tree node identified by the specified item path.
ExpandAll	Expands all tree nodes.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemRect	Returns the size and position of the item relative to the tree.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsExpanded	Returns 'true' if the specified item path is expanded.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsPathEditable	whether an item path is editable.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollPathIntoView	Scrolls the given item path to a visible position.
Select	Selects a tree node identified by the specified item path.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
StartEditing	Starts editing at given item path or current selection.
StopEditing	Stops the current editing session.

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JVerticalScrollBar Class

Description
Represents a vertical scroll bar.

Inheritance Hierarchy

- *JScrollBar*
  - JVerticalScrollBar

Syntax

```
'Declaration
Public Class JVerticalScrollBar _
Inherits JScrollBar
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	the block increment value of the scroll bar. (Inherited from *IAWTScroller*)
Position	the current position of the scroll bar. (Inherited from *IAWTScroller*)
Range	the range value of the scroll bar. (Inherited from *IAWTScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IAWTScroller*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
Page	Increments the current scroll position by specified number of pages. (Inherited from *IAWTScroller*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IAWTScroller*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Step	Increments the current scroll position by specified steps. (Inherited from *IAWTScroller*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JViewport Class

Description
Represents the Swing viewport.

Inheritance Hierarchy

- *JComponent*

- JViewport

Syntax

```
'Declaration
Public Class JViewport _
Inherits JComponent
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
ToolTipText	the tool tip text of a Swing component. (Inherited from *JComponent*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
ViewRectangle	the visible part of the view, in view coordinates.
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An

Name	Description
	exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

JWindow Class

Description
Represents a container that can be displayed anywhere on the user's desktop.

Inheritance Hierarchy

- *AWTWindow*
  - JWindow

Syntax
```
'Declaration
Public Class JWindow _
Inherits AWTWindow
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)

Name	Description
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsApplication Class

Description
Represents an Oracle Forms application.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsApplication

Syntax

```
'Declaration
Public Class OracleFormsApplication _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SplashScreenRunning	whether the application currently displays a splash screen.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsButton Class

Description
Represents a push button.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsButton

Syntax

```
'Declaration
Public Class OracleFormsButton _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Default	Whether the button is the default button.
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)

Name	Description
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the

Name	Description
	timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsCheckbox Class

Description
Represents a check box.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsCheckbox

Syntax

```
'Declaration
Public Class OracleFormsCheckbox _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Checked	whether the check box is checked.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the check box.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)

Name	Description
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsChoice Class

Description
Represents a choice.

Inheritance Hierarchy

- *OracleFormsContainer*

- OracleFormsChoice
  - *OracleFormsComboBox*
  - *OracleFormsListBox*
  - *OracleFormsPopList*

Syntax

```
'Declaration
Public Class OracleFormsChoice _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
ItemCount	the number of items in the choice.
Items	the list of choice items.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	the index of the selected item. -1 (4Test: 0) is returned, if no item is currently selected.
SelectedItem	the selected item.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemText	Returns the item at the specified index.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the specified item in the choice.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsComboBox Class

Description
Represents a combo box.

Inheritance Hierarchy

- *OracleFormsChoice*
  - OracleFormsComboBox

Syntax

```
'Declaration
Public Class OracleFormsComboBox _
Inherits OracleFormsChoice
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
DropDownOpen	whether the drop down is currently open.
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
ItemCount	the number of items in the choice. (Inherited from *OracleFormsChoice*)
Items	the list of choice items. (Inherited from *OracleFormsChoice*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	the index of the selected item. -1 (4Test: 0) is returned, if no item is currently selected. (Inherited from *OracleFormsChoice*)
SelectedItem	the selected item. (Inherited from *OracleFormsChoice*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetItemText	Returns the item at the specified index. (Inherited from *OracleFormsChoice*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the specified item in the choice. (Inherited from *OracleFormsChoice*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetText	Sets the text of the combo box to the specified text.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsContainer Class

Description
Represents a component that can contain other Oracle Forms components.

Inheritance Hierarchy

- *AWTContainer*
  - OracleFormsContainer
    - *OracleFormsApplication*
    - *OracleFormsButton*
    - *OracleFormsCheckbox*
    - *OracleFormsChoice*
    - *OracleFormsLabel*
    - *OracleFormsListView*
    - *OracleFormsMenu*
    - *OracleFormsMenuItem*

- *OracleFormsRadioButton*
- *OracleFormsScrollbar*
- *OracleFormsStatusArea*
- *OracleFormsStatusBar*
- *OracleFormsStatusBarItem*
- *OracleFormsStatusIndicator*
- *OracleFormsTabBar*
- *OracleFormsTabBarItem*
- *OracleFormsTabPanel*
- *OracleFormsTextField*
- *OracleFormsTitleBar*
- *OracleFormsToolBar*
- *OracleFormsToolBarItem*
- *OracleFormsTree*

Syntax

```
'Declaration
Public Class OracleFormsContainer _
Inherits AWTContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsHorizontalScrollbar Class

Description
Represents a horizontal scroll bar.

Inheritance Hierarchy

- *OracleFormsScrollbar*
  - OracleFormsHorizontalScrollbar

Syntax

```
'Declaration
Public Class OracleFormsHorizontalScrollbar _
Inherits OracleFormsScrollbar
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScroller*)
Position	the current position of the scroll bar. (Inherited from *IOracleFormsScroller*)
Range	the range value of the scroll bar. (Inherited from *IOracleFormsScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScroller*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding.

Name	Description
	In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
Page	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScroller*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScroller*)
ScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScroller*)
ScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScroller*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Step	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScroller*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsLabel Class

Description
Represents a class for placing text in a container.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsLabel

Syntax

```
'Declaration
Public Class OracleFormsLabel _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsListBox Class

Description
Represents a ListBox.

Inheritance Hierarchy

- *OracleFormsChoice*
  - OracleFormsListBox

Syntax

```
'Declaration
Public Class OracleFormsListBox _
Inherits OracleFormsChoice _
Implements IOracleFormsScrollable
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)

Name	Description
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
HorizontalScrollbarPageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarPosition	the current position of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarRange	the range value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarStepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
ItemCount	the number of items in the choice. (Inherited from *OracleFormsChoice*)
Items	the list of choice items. (Inherited from *OracleFormsChoice*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	the index of the selected item. -1 (4Test: 0) is returned, if no item is currently selected. (Inherited from *OracleFormsChoice*)
SelectedItem	the selected item. (Inherited from *OracleFormsChoice*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticalScrollbarPageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarPosition	the current position of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarRange	the range value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarStepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
ClickItem	Clicks the specified list item.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleClickItem	Double-clicks a list item.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalScrollbar	Gets the horizontal scroll bar. (Inherited from *IOracleFormsScrollable*)
GetItemText	Returns the item at the specified index. (Inherited from *OracleFormsChoice*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalScrollbar	Gets the vertical scroll bar. (Inherited from *IOracleFormsScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
HorizontalScrollbarPage	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarStep	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScrollable*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the specified item in the choice. (Inherited from *OracleFormsChoice*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
VerticalScrollbarPage	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarStep	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScrollable*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsListView Class

Description
Represents a listview.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsListView

Syntax

```
'Declaration
Public Class OracleFormsListView _
Inherits OracleFormsContainer _
Implements IOracleFormsScrollable
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
AllCellTexts	the text values of all cells in the listview. The cells are grouped in rows, where each row contains the texts of its cells.
Background	The background color of the component. (Inherited from *AWTComponent*)
ColumnCount	the number of columns in the listview.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)

Name	Description
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
FocusedRow	the index of the focused row, -1 (4Test: 0) is returned if no row is focused.
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
HorizontalScrollbarPageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarPosition	the current position of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarRange	the range value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarStepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
RowCount	the number of rows in the listview.
SelectedRow	the index of the selected row, -1 (4Test: 0) is returned if no row is selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticalScrollbarPageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarPosition	the current position of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarRange	the range value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarStepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearSelection	Clears the selection.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCellText	Returns the cell text.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetColumnName	Returns the name of the column at the specified index.
GetColumnWidth	Returns the width of the specified column.
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalScrollbar	Gets the horizontal scroll bar. (Inherited from *IOracleFormsScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalScrollbar	Gets the vertical scroll bar. (Inherited from *IOracleFormsScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
HorizontalScrollbarPage	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarStep	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScrollable*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsRowVisible	Returns whether the specified row is visible.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollToRow	Scrolls to the specified row.
SelectRow	Selects the specified row.
SetColumnWidth	Sets the width of the specified column.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
VerticalScrollbarPage	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarStep	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScrollable*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout

Name	Description
	parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsMenu Class

Description
Represents a top-level menu.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsMenu
    - *OracleFormsPopupMenu*

Syntax

```
'Declaration
Public Class OracleFormsMenu _
Inherits OracleFormsContainer _
Implements IOracleFormsMenuBase
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)

Name	Description
Checked	Whether the MenuItem is checked. (Inherited from *IOracleFormsMenuBase*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks a menu item
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsChecked	a value that indicates whether the MenuItem is checked.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects a menu item
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks a menu item
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use

Name	Description
	WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsMenuItem Class

Description
Represents a selectable item inside a Menu.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsMenuItem

Syntax

```
'Declaration
Public Class OracleFormsMenuItem _
Inherits OracleFormsContainer _
Implements IOracleFormsMenuBase
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Checked	Whether the MenuItem is checked. (Inherited from *IOracleFormsMenuBase*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)

Name	Description
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks a menu item
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects a menu item
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks a menu item
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the

Name	Description
	optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsPopList Class

Description
Represents a PopList.

Inheritance Hierarchy

- *OracleFormsChoice*
  - OracleFormsPopList

Syntax

```
'Declaration
Public Class OracleFormsPopList _
Inherits OracleFormsChoice
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
DropDownOpen	whether the drop down is currently open.
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
ItemCount	the number of items in the choice. (Inherited from *OracleFormsChoice*)
Items	the list of choice items. (Inherited from *OracleFormsChoice*)
Name	The name of the component. (Inherited from *AWTComponent*)

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	the index of the selected item. -1 (4Test: 0) is returned, if no item is currently selected. (Inherited from *OracleFormsChoice*)
SelectedItem	the selected item. (Inherited from *OracleFormsChoice*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)

Name	Description
GetItemText	Returns the item at the specified index. (Inherited from *OracleFormsChoice*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)

Name	Description
Select	Selects the specified item in the choice. (Inherited from *OracleFormsChoice*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsPopupMenu Class

Description
Represents a context menu.

Inheritance Hierarchy

- *OracleFormsMenu*
  - OracleFormsPopupMenu

Syntax

```
'Declaration
Public Class OracleFormsPopupMenu _
Inherits OracleFormsMenu
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Checked	Whether the MenuItem is checked. (Inherited from *IOracleFormsMenuBase*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks a menu item (Inherited from *OracleFormsMenu*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)

Name	Description
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsChecked	a value that indicates whether the MenuItem is checked. (Inherited from *OracleFormsMenu*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects a menu item (Inherited from *OracleFormsMenu*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks a menu item (Inherited from *OracleFormsMenu*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsRadioButton Class

Description
Represents a radio button.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsRadioButton

Syntax

```
'Declaration
Public Class OracleFormsRadioButton _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Checked	Whether the radio button is checked.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks the radio button.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsScrollbar Class

Description
Represents a scroll bar.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsScrollbar
    - *OracleFormsHorizontalScrollbar*
    - *OracleFormsVerticalScrollbar*

Syntax

```
'Declaration
Public Class OracleFormsScrollbar _
Inherits OracleFormsContainer _
Implements IOracleFormsScroller
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)

Name	Description
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScroller*)
Position	the current position of the scroll bar. (Inherited from *IOracleFormsScroller*)
Range	the range value of the scroll bar. (Inherited from *IOracleFormsScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScroller*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)

Name	Description
Page	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScroller*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScroller*)
ScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScroller*)
ScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScroller*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Step	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScroller*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsStatusArea Class

Description
Represents the area containing status information.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsStatusArea

Syntax

```
'Declaration
Public Class OracleFormsStatusArea _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)

Name	Description
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsStatusBar Class

Description
Represents a status bar.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsStatusBar

Syntax

```
'Declaration
Public Class OracleFormsStatusBar _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)

Name	Description
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsStatusBarItem Class

Description
Represents an item within a status bar.

Inheritance Hierarchy

- *OracleFormsContainer*

- `OracleFormsStatusBarItem`

Syntax

```
'Declaration
Public Class OracleFormsStatusBarItem _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)

Name	Description
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsStatusIndicator Class

Description
Represents a status indicator.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsStatusIndicator

Syntax

```
'Declaration
Public Class OracleFormsStatusIndicator _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)

Name	Description
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsTabBar Class

Description
Represents a tab bar.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsTabBar

Syntax

```
'Declaration
Public Class OracleFormsTabBar _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedTab	the currently selected tab.
TabCount	the number of tabs.
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetTabIndex	Returns the tab index.
GetTabTitle	Returns the tab title.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsTabEnabled	Returns whether or not the tab is currently enabled.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the specified tab.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)

Name	Description
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsTabBarItem Class

Description
Represents a tab bar item.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsTabBarItem

Syntax

```
'Declaration
Public Class OracleFormsTabBarItem _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the tab.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)

Name	Description
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsTabPanel Class

Description
Represents a container class for components.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsTabPanel

Syntax

```
'Declaration
Public Class OracleFormsTabPanel _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsTextField Class

Description
Represents an Oracle Forms text field.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsTextField

Syntax

```
'Declaration
Public Class OracleFormsTextField _
Inherits OracleFormsContainer _
Implements IOracleFormsScrollable
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
CaretPosition	the position of the text insertion caret for the text field.
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Editable	whether the text of the text field can be changed by the user.

Name	Description
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
HorizontalScrollbarPageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarPosition	the current position of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarRange	the range value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarStepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
IsPassword	whether the text field is a password text field.
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PasswordChar	the character that is displayed instead of the characters that are actually typed, if the text field is a password text field.
SelectedText	the current selected text.
SelectionEnd	the end position of the selected text.
SelectionStart	the start position of the selected text.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticalScrollbarPageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarPosition	the current position of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarRange	the range value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarStepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Clear	Removes all the text from the text field.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalScrollbar	Gets the horizontal scroll bar. (Inherited from *IOracleFormsScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalScrollbar	Gets the vertical scroll bar. (Inherited from *IOracleFormsScrollable*)

Name	Description
HighlightObject	Highlights this object. (Inherited from *TestObject*)
HorizontalScrollbarPage	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarStep	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScrollable*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)

Name	Description
SelectAll	Selects all the text in the text field.
SetCaretPosition	Sets the position of the text insertion caret for this text field.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetSelectionRange	Selects the text within the specified range.
SetText	Sets the text of the text field to the specified text.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
VerticalScrollbarPage	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarStep	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScrollable*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsTitleBar Class

Description
Represents a title bar.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsTitleBar

Syntax

```
'Declaration
Public Class OracleFormsTitleBar _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)

Name	Description
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsToolBar Class

Description
Represents a toolbar.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsToolBar

Syntax

```
'Declaration
Public Class OracleFormsToolBar _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsToolBarItem Class

Description
Represents an element in a toolbar.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsToolBarItem

Syntax

```
'Declaration
Public Class OracleFormsToolBarItem _
Inherits OracleFormsContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)

Name	Description
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
Select	Selects the element.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any

Name	Description
	results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsTree Class

Description
Represents a tree.

Inheritance Hierarchy

- *OracleFormsContainer*
  - OracleFormsTree

Syntax

```
'Declaration
Public Class OracleFormsTree _
Inherits OracleFormsContainer _
Implements IOracleFormsScrollable
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)

Name	Description
Height	The current height of the component. (Inherited from *AWTComponent*)
HorizontalScrollbarPageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarPosition	the current position of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarRange	the range value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarStepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedItemPath	the list of selected item path.
SelectedItemText	the selected item text.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
VerticalScrollbarPageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarPosition	the current position of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarRange	the range value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarStepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScrollable*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
ClickItem	Clicks on a tree node specified by the item path.

Name	Description
Collapse	Collapses the tree node identified by the specified item path.
CollapseAll	Collapses all tree nodes.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleClickItem	Double-clicks on a tree node specified by the item path.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Expand	Expands the tree node identified by the specified item path.
ExpandAll	Expands all tree nodes.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalScrollbar	Gets the horizontal scroll bar. (Inherited from *IOracleFormsScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalScrollbar	Gets the vertical scroll bar. (Inherited from *IOracleFormsScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
HorizontalScrollbarPage	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScrollable*)

Name	Description
HorizontalScrollbarScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScrollable*)
HorizontalScrollbarStep	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScrollable*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsExpanded	Returns 'true' if the specified item path is expanded.
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
IsVisible	Returns 'true' if the specified item path is visible.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollItemIntoView	Scrolls the given item path to a visible position.
Select	Selects a tree node identified by the specified item path.

Name	Description
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
VerticalScrollbarPage	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScrollable*)
VerticalScrollbarStep	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScrollable*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

OracleFormsVerticalScrollbar Class

Description
Represents a vertical scroll bar.

Inheritance Hierarchy

- *OracleFormsScrollbar*
  - OracleFormsVerticalScrollbar

Syntax

```
'Declaration
Public Class OracleFormsVerticalScrollbar _
Inherits OracleFormsScrollbar
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)

Name	Description
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
PageSize	the block increment value of the scroll bar. (Inherited from *IOracleFormsScroller*)
Position	the current position of the scroll bar. (Inherited from *IOracleFormsScroller*)
Range	the range value of the scroll bar. (Inherited from *IOracleFormsScroller*)
StepSize	the unit increment value of the scroll bar. (Inherited from *IOracleFormsScroller*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
Page	Increments the current scroll position by specified number of pages. (Inherited from *IOracleFormsScroller*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
ScrollTo	Scrolls to the specified position. (Inherited from *IOracleFormsScroller*)
ScrollToMax	Scrolls to the maximum position of the scroller. (Inherited from *IOracleFormsScroller*)
ScrollToMin	Scrolls to the minimum position of the scroller. (Inherited from *IOracleFormsScroller*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Step	Increments the current scroll position by specified steps. (Inherited from *IOracleFormsScroller*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by

Name	Description
	setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

SplitPaneDivider Class

Description
Represents a split pane divider.

Inheritance Hierarchy

- *AWTContainer*
  - SplitPaneDivider

Syntax

```
'Declaration
Public Class SplitPaneDivider _
Inherits AWTContainer
```

Properties

Name	Description
AccessibleName	The accessible name of the component. (Inherited from *AWTComponent*)
AccessibleRole	The accessible role of the component. (Inherited from *AWTComponent*)
Background	The background color of the component. (Inherited from *AWTComponent*)
Cursor	The name of the cursor set in the component. (Inherited from *AWTComponent*)
Enabled	whether the component is enabled. (Inherited from *AWTComponent*)
Focusable	whether the component can be focused. (Inherited from *AWTComponent*)
Font	The font of the component. (Inherited from *AWTComponent*)

Name	Description
Foreground	The foreground color of the component. (Inherited from *AWTComponent*)
Height	The current height of the component. (Inherited from *AWTComponent*)
Name	The name of the component. (Inherited from *AWTComponent*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Width	The current width of the component. (Inherited from *AWTComponent*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *AWTComponent*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ScrollIntoView	Scrolls the component into view if it is hidden and a child of a scroll pane. (Inherited from *AWTComponent*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Win32 Class Reference

Lists the available classes for testing Windows API-based controls.

AccessibleControl Class

Description

The class controls identified by the Accessibility extension.

Inheritance Hierarchy

- *BaseGuiTestObject*
  - AccessibleControl

Syntax

```
'Declaration
Public Class AccessibleControl _
Inherits BaseGuiTestObject
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Description	The description text of the accessible object.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Role	The role of this accessible object.
State	The state string that describes the different states of this accessible object.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoDefaultAction	Executes the default action of the underlying accessible control supports a default action.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)

Name	Description
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Header Class

Description
The class for list header controls. Headers are most commonly used in controls of the ListViewW32 class.

Inheritance Hierarchy

- *Control*
  - Header

Syntax

```
'Declaration
Public Class Header _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
ColumnCount	The number of columns in the header.
ColumnOrder	The header's item indices as they are currently ordered.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DividerDoubleSelect	Double-clicks the divider on the right side of the specified item.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Double-clicks an item.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the

Name	Description
	agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetColumnTitle	Returns the title of a specified header item.
GetColumnWidth	Returns the width of a specified column.
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Resize	Resizes an item of the header.
Select	Selects an item or button.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ListView Class

Description

The class for elaborated list controls that support multiple ways of viewing a list of nonhierarchical data. These views include large and small rearrangable icons and a detail (or report) view.

Inheritance Hierarchy

- *Control*
  - ListView

Syntax

```
'Declaration
Public Class ListView _
Inherits Control
```

Properties

Name	Description
AllowsCheck	Whether this list view supports checking items.
AllowsMultiSelect	Whether the list view supports selecting multiple items.
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
CheckedIndices	A list containing the indices of the checked items.
CheckedItems	A list containing the checked items.

Name	Description
ColumnCount	The number of columns per item.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items contained in the list view.
Items	A list containing all items of the listview.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndices	A list containing the indices of the selected items.
SelectedItems	A list containing the selected items.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Checks an item.
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Double-clicks an item.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
ExtendSelect	Selects a range of items by extending the selection in the extend-selection listview.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetContents	Returns the contents of a listview control. Each returned item is delimited by a semicolon. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetItemImageIndex	Returns the index into the image list of a list view that contains icons. This allows to verify that the correct image is used for an item.
GetItemIndex	Returns the index of the list view item or -1 if is not found.
GetItemRect	Returns rectangle of the item. If getIconRect is set then the rectangle of the icon of the item will be returned otherwise the rectangle of the item text is returned. If a columnIndex is specified the rectangle of this column is returned.
GetItemText	Returns the text of an item.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MultiSelect	Adds an item in the list to the set of currently selected items.
MultiUnselect	Removes an item in the list from the set of selected items.
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetItemFocus	Gives focus to the item in the list.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks an item.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

MonthCalendar Class

Description
The class that selects the calendar date and time.

Inheritance Hierarchy

- *Control*
  - MonthCalendar

Syntax

```
'Declaration
Public Class MonthCalendar _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
ExtendSelect	Selects a range of dates and times.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
OpenMonthMenu	Opens a calendar menu to a specific month.

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects a date and time.
SelectNextMonth	Selects the next month.
SelectPreviousMonth	Selects the previous month.
SelectToday	Selects the date for today.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use

Name	Description
	WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Pager Class

Description

The class for tabbed, multi-page dialogs and for button bars. On Windows, this class corresponds to the TabControl control. On Windows, Silk Test only supports major tabs; it does not support minor tabs (also called 'subpages' or 'child pages').

Inheritance Hierarchy

- *Control*
  - Pager

Syntax

```
'Declaration
Public Class Pager _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
Horizontal	Whether the up/down control is horizontal.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)

Name	Description
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollDown	Moves the scroll bar down.
ScrollUp	Moves the scroll bar up.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ProgressBar Class

Description
The class for progress bar controls. A progress bar displays a progress.

Inheritance Hierarchy

- *Control*
  - ProgressBar

Syntax

```
'Declaration
Public Class ProgressBar _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	The current position.
Range	The range of the progress bar control.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

StatusBar Class

Description
The class for status bar controls. A status bar is a container for a set of text labels that change dynamically.

Inheritance Hierarchy

- *Control*
  - StatusBar

Syntax
```
'Declaration
Public Class StatusBar _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)

Name	Description
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

UpDown Class

Description
The class for up/down controls.

Inheritance Hierarchy

- *Control*
  - UpDown
    - *NumericUpDown*

Syntax

```
'Declaration
Public Class UpDown _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
Horizontal	Whether the up/down control is horizontal.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	The position for the up/down control.
Range	The range of the up/down control.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
WrapAround	Whether values are wrapped if smaller than minimum or larger than maximum.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Decrement	Decreases the position value of the up/down control.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Increment	Increases the position value of the up/down control.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Windows Forms Class Reference

Lists the available classes for testing Windows Forms controls.

CheckedListBox Class

Description
CheckedListBox represents list boxes that contain a check box.

Inheritance Hierarchy

- *ListBox*
  - CheckedListBox

Syntax

```
'Declaration
Public Class CheckedListBox _
Inherits ListBox
```

Properties

Name	Description
AllowsMultiSelect	Whether the control supports selecting multiple items. (Inherited from *ListBox*)
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
CheckedIndices	The indices of the checked item(s).
CheckedItems	The names of the checked item(s).
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items in the list box. (Inherited from *ListBox*)
Items	A list of items in the list box. (Inherited from *ListBox*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The index of the first selected item. (Inherited from *ListBox*)
SelectedIndices	The indices of the selected item(s). (Inherited from *ListBox*)
SelectedItem	The name of the first selected item. (Inherited from *ListBox*)
SelectedItems	The names of the selected item(s). (Inherited from *ListBox*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Check	Selects an item from the checked list box.
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelect	Double-clicks an item in the list box. (Inherited from *ListBox*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
ExtendSelect	Selects a range of items by extending the selection in the list box. (Inherited from *ListBox*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MultiSelect	Selects an item in the multi- or extend-selection list box. (Inherited from *ListBox*)
MultiUnselect	Unselects an item in the multi- or extend-selection list box. (Inherited from *ListBox*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item from the list box. (Inherited from *ListBox*)
SelectRange	Selects a range of items in the extend-selection list box. (Inherited from *ListBox*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Selects an item from the checked list box.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DataGrid Class

Description
The class for datagrids and data grid view.

Inheritance Hierarchy

- *Control*
  - DataGrid

Syntax

```
'Declaration
Public Class DataGrid _
Inherits Control
```

Properties

Name	Description
AllowsMultiSelect	Whether the table supports selecting multiple items.
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
ColumnCount	The number of columns in the DataGrid.
ColumnItems	A list of all items in the column.
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
Items	A list of all items in the column.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
RowCount	The number of rows in the DataGrid.
RowItems	A list of all items in the row.
SelectedItems	The selected item(s).
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
DoubleSelectItem	Double-clicks an item.

Name	Description
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
ExtendSelectItem	Expands the selection to the given item.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MultiSelectItem	Adds a row from the table to the set of selected rows.
MultiUnselectItem	Removes a item from the set of selected items.
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Tries to bring the item to the top left corner of the datagrid.
SelectAllItems	Selects all items in the datagrid.
SelectItem	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)

Name	Description
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DataGridColumn Class

Description
The class for columns in a table.

Inheritance Hierarchy

- *Item*
  - DataGridColumn

Syntax
```
'Declaration
Public Class DataGridColumn _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)

Name	Description
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
Width	The width of the column.

Methods

Name	Description
AutoResize	Automatically resizes a column by double clicking its divider.
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Resize	Resizes a column.

Name	Description
ScrollIntoView	Tries to bring the column to the left by scrolling the data grid.
Select	Selects a column.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DataGridItem Class

Description
The class for datagrids and data grid view rows.

Inheritance Hierarchy

- *Item*
  - DataGridItem

Syntax
```
'Declaration
Public Class DataGridItem _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Column	The column of the item in the data grid
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Row	The row of the item in the data grid
Selected	Whether the item is selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha

Name	Description
	channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)

Name	Description
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Tries to bring the item to the top left corner of the datagrid.
Select	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DataGridRow Class

Description
The class for rows in a table.

Inheritance Hierarchy

- *Item*
  - DataGridRow

Syntax

```
'Declaration
Public Class DataGridRow _
Inherits Item
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)

Name	Description
Height	The width of the column.
ItemCount	The number of items in the row.
Items	A list of items in the row.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Resize	Resizes a column.
ScrollIntoView	Tries to bring the row to the top by scrolling the data grid.

Name	Description
Select	Selects a column.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomainUpDown Class

Description
The class for up/down controls.

Inheritance Hierarchy

- *Control*
  - DomainUpDown

Syntax
```
'Declaration
Public Class DomainUpDown _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
ItemCount	The number of items contained in the list view.
Items	The items in the domain up/down control.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
SelectedIndex	The position for the up/down control.
SelectedItem	The position for the up/down control.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
WrapAround	Whether values are wrapped if smaller than minimum or larger than maximum.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Decrement	Decreases the position value of the up/down control.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)

Name	Description
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Increment	Increases the position value of the up/down control.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Select	Selects an item.
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)

Name	Description
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

ElementHost Class

Description
A Windows Forms control that can be used to host a Windows Presentation Foundation (WPF) element.

Inheritance Hierarchy

- *Control*
  - ElementHost

Syntax

```
'Declaration
Public Class ElementHost _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)

Name	Description
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive

Name	Description
	random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FormsHost Class

Description
An element that allows you to host a Windows Forms control on a WPF page.

Inheritance Hierarchy

- *Control*
  - FormsHost

Syntax
```
'Declaration
Public Class FormsHost _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)

Name	Description
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)

Name	Description
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

FormsWindow Class

Description
FormsWindow is the class that represents the top-level window of a Windows Forms application.

Inheritance Hierarchy
- *Window*
  - FormsWindow

Syntax
```
'Declaration
Public Class FormsWindow _
Inherits Window
```

Properties

Name	Description
Application	The name of the Application that this Window belongs to. (Inherited from *Window*)
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)

Name	Description
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
LoadAssembly	Loads the assembly on the specified path in the Windows Forms application.
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
SetActive	Makes the window active. (Inherited from *IMoveable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)

Name	Description
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

MenuStrip Class

Description
MenuStrip is the class that creates a new menu strip for a Windows Forms application.

Inheritance Hierarchy

- *Control*
  - MenuStrip

Syntax

```
'Declaration
Public Class MenuStrip _
Inherits Control
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

NumericUpDown Class

Description
The class for numeric up/down controls.

Inheritance Hierarchy

- *UpDown*
  - NumericUpDown

Syntax

```
'Declaration
Public Class NumericUpDown _
Inherits UpDown
```

Properties

Name	Description
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
Horizontal	Whether the up/down control is horizontal. (Inherited from *UpDown*)
Maximum	The maximum value of the up/down control.
Minimum	The minimum value of the up/down control.
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Position	The position for the up/down control. (Inherited from *UpDown*)
PositionDecimal	The position for the up/down control.
Range	The range of the up/down control. (Inherited from *UpDown*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)

Name	Description
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
WrapAround	Whether values are wrapped if smaller than minimum or larger than maximum. (Inherited from *UpDown*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
Decrement	Decreases the position value of the up/down control. (Inherited from *UpDown*)
DecrementDecimal	Decreases the position value of the up/down control.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)

Name	Description
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Increment	Increases the position value of the up/down control. (Inherited from *UpDown*)
IncrementDecimal	Increases the position value of the up/down control.
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

xBrowser Class Reference

Lists the available classes for testing xBrowser controls.

BrowserApplication Class

Description

Represents the top-level window of a browser application.

Note: `BrowserApplication` is derived from `BaseGuiTestObject` and inherits the two scrollbar methods `GetHorizontalBar` and `GetVerticalBar`. However, these two methods do not work on a `BrowserApplication`.

Inheritance Hierarchy

- *Window*
  - `BrowserApplication`

Syntax

```
'Declaration
Public Class BrowserApplication _
Inherits Window
```

Properties

Name	Description
Application	The name of the Application that this Window belongs to. (Inherited from *Window*)
Background	The background color of the GUI object. (Inherited from *GuiTestObject*)
Enabled	Whether the GUI object is enabled. (Inherited from *GuiTestObject*)
Font	The font type of the GUI object. (Inherited from *GuiTestObject*)
Foreground	The foreground color of the GUI object. (Inherited from *GuiTestObject*)
NativeHandle	The native window handle for the object. (Inherited from *INativeWindow*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *GuiTestObject*)
WindowState	The state of a window. Values include: 1=minimized, 2=maximized, 3=restored (Inherited from *IMoveable*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearCache	Clears the browser's cache.
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Closes the window. (Inherited from *IMoveable*)
CloseOtherTabs	Closes all tabs except the active one. When several tabs are open, Internet Explorer only displays some of the tabs and scroll buttons. This method closes all other tabs except for the selected one, including the tabs that are not currently visible. This method is not supported for mobile Web applications.
CloseSynchron	Closes the window and waits until the window is closed. (Inherited from *IMoveable*)
CloseTab	Closes the specified tab. If no tab is specified, the active tab is closed. When several tabs are open, Internet Explorer only displays some of the tabs and scroll buttons. As a result, only the currently visible tabs work with this method, which affects index semantics. For example, only the currently visible tabs can be closed. Index 0 identifies the first visible tab. This method is not supported for mobile Web applications.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
ExistsTab	Returns true if the tab exists and false otherwise. This method is not supported for mobile Web applications.
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetActiveTab	Returns the active tab. This method is not supported for mobile Web applications.

Name	Description
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCloseWindows	Returns the windows that need to be closed in order so that only the main window of the application is open. (Inherited from *IMoveable*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetFocus	Returns the object with the input focus. (Inherited from *IMoveable*)
GetHorizontalBar	Returns the horizontal scroll bar for this control. (Inherited from *IScrollable*)
GetNextCloseWindow	Returns the next window that need to be closed in order to close all windows of the application except the main window. (Inherited from *IMoveable*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetSelectedTab	Returns the currently selected tab. This method is not supported for mobile Web applications.
GetSelectedTabIndex	Returns the index of the active tab. When several tabs are open, Internet Explorer only displays some of the tabs and scroll buttons. As a result, this method only returns the index relative to the currently visible tabs. Index 0 identifies the first visible tab. This method is not supported for mobile Web applications.
GetSelectedTabName	Returns the name of the active tab. This method is not supported for mobile Web applications.
GetTabCount	Returns the number of open tabs. When several tabs are open, Internet Explorer only displays some of the tabs and scroll buttons. As a result, this method only returns the number of currently visible tabs. This method is not supported for mobile Web applications.
GetVerticalBar	Returns the vertical scroll bar for this control. (Inherited from *IScrollable*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)

Name	Description
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsActive	Returns whether the window is set to active. (Inherited from *IMoveable*)
IsFocused	Return whether the control has focus. (Inherited from *IFocusable*)
Maximize	Maximizes the window. (Inherited from *IMoveable*)
Minimize	Reduces the window to an icon. (Inherited from *IMoveable*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Move	Moves the window. (Inherited from *IMoveable*)
Navigate	Navigate to the specified URL in the currently active tab/window
OpenContextMenu	Opens a context menu at the specified position. (Inherited from *BaseGuiTestObject*)
OpenTab	Opens a new tab and navigates to the given URL. If no URL is given, the tab navigates to about:blank. This method is not supported for mobile Web applications.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Restore	Restores the window to its previous size. (Inherited from *IMoveable*)
SelectTab	Selects the specified tab. When several tabs are open, Internet Explorer only displays some of the tabs and scroll buttons. As a result, only the currently visible tabs work with this method, which affects index semantics. For example, only the currently visible tabs can be selected. Index 0 selects the first visible tab. This method is not supported for mobile Web applications.
SetActive	Makes the window active. (Inherited from *IMoveable*)

Name	Description
SetFocus	Gives focus to the control. (Inherited from *IFocusable*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Size	Resizes the window. (Inherited from *IMoveable*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

BrowserObject Class

Description
Base class for all objects that are contained within a BrowserApplication.

Inheritance Hierarchy

- *TestObject*
  - BrowserObject
    - *BrowserWindow*
    - *DomElement*

Syntax
```
'Declaration
Public Class BrowserObject _
Inherits TestObject _
Implements IClickable, IKeyable
```

Properties

Name	Description
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay.

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is

Name	Description
	specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
SetFocus	Gives focus to the control.
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

BrowserWindow Class

Description
A browser window is the top-level container for a web page. It exposes the DOM tree through DomElements. E.g. a tab in IE7, an embedded browser control in SWT.

Inheritance Hierarchy
- *BrowserObject*
  - BrowserWindow

Syntax
```
'Declaration
Public Class BrowserWindow _
Inherits BrowserObject
```

Properties
Name	Description
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
ViewportHeight	The height of the browser viewport
ViewportName	The name of the viewport that matches the innerWidth and innerHeight of the browser window
ViewportOrientation	The orientation of the browser viewport, this can be portrait or landscape
ViewportWidth	The width of the browser viewport
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods
Name	Description
AcceptAlert	Accepts a JavaScript alert by pressing OK.

Name	Description
Back	Goes back in the history.
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
CaptureFullPageBitmap	Capture a screenshot of the whole web page and save it as a file to the specified path. The filename extension determines the file type. Possible extensions are .jpg, .jpeg, .png, and .bmp.
Click	Clicks on the object. (Inherited from *IClickable*)
Close	Close a modal browser dialog, a tab of the browser, or the browser itself if it is the last tab (or IE6).
DismissAlert	Dismisses a JavaScript alert.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates JavaScript code within the top-level document context.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
Forward	Goes forward in the history.
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetAlertText	Retrieves the message text of a JavaScript alert.
GetBrowserMajorVersion	Returns the browser's major version number.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of the BrowserWindow in HTML pixels. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser

Name	Description
	capable and your position calculations are not affected by changing zoom levels on mobile browsers.
GetPageSource	Returns the page source of the web site
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetUrl	Returns the URL for the main document.
GetUserAgent	Returns the browser's user agent string
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsAlertPresent	Returns true if a JavaScript alert is open
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Navigate	Navigates the main document to an URL.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)

Name	Description
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
Reload	Reloads the main document as the reload button in the browser does.
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetViewportName	Set the size of the browser window to the specified size from your Browser Size list.
SetViewportSize	Sets the size of the browser window to the specified width and height.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomButton Class

Description

A DomButton represents all DOM elements that were specified using one of the following tags: <input type='submit'>, <input type='reset'>, <input type='button'> or <button>.

Inheritance Hierarchy

- *DomElement*
  - DomButton

Syntax

```
'Declaration
Public Class DomButton _
Inherits DomElement
```

Properties

Name	Description
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)

Name	Description
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position

Name	Description
	calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
Select	Clicks a button.
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)

Name	Description
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomCheckBox Class

Description
A DomCheckBox represents all DOM elements that were specified using <input type='checkbox'> tag.

Inheritance Hierarchy

- *DomElement*
  - DomCheckBox

Syntax
```
'Declaration
Public Class DomCheckBox _
Inherits DomElement
```

Properties

Name	Description
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
State	The state of a check box. Values include: 1=checked,2=unchecked, 3=undecided
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)

Name	Description
Check	Checks the check box.
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by

Name	Description
	GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
IsChecked	Returns *true* if the checkbox is checked and *false* otherwise.
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
Select	Selects the check box
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetUndecided	Sets the check box to the undecided state.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
Toggle	Toggles the state of the check box. If the checkbox is checked it is unchecked and vice versa. If the checkbox is in the undecided state it is left undecided.
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Uncheck	Unchecks the check box.
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Name	Description
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomElement Class

Description
A DomElement represents any element in the DOM tree.

Inheritance Hierarchy
- *BrowserObject*
  - DomElement
    - *DomButton*
    - *DomCheckBox*
    - *DomEmbeddedElement*
    - *DomForm*
    - *DomLink*
    - *DomListBox*
    - *DomRadioButton*
    - *DomTable*
    - *DomTableRow*
    - *DomTextField*

Syntax
```
'Declaration
Public Class DomElement _
Inherits BrowserObject
```

Properties
Name	Description
IsFocused	Whether the DOM element has focus.
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses.
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications.
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications.
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context.
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name.
GetDomAttribute	Gets the value of an object specific DOM attribute.
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications.
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers.
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element.
Highlight	Highlights a DOM element.
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)

Name	Description
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window.
SetDomAttribute	Sets the value of an object specific DOM attribute.
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout

Name	Description
	parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomEmbeddedElement Class

Description
A DOM element that was specified by an <embed> or <object> tag.

Inheritance Hierarchy

- *DomElement*
  - DomEmbeddedElement

Syntax
```
'Declaration
Public Class DomEmbeddedElement _
Inherits DomElement
```

Properties

Name	Description
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)

Name	Description
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomForm Class

Description

A DomForm represents all DOM elements that were specified using the <form> tag. All methods and properties in this class are not supported for mobile Web applications.

Inheritance Hierarchy

- *DomElement*
  - DomForm

Syntax

```
'Declaration
Public Class DomForm _
Inherits DomElement
```

Properties

Name	Description
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)

Name	Description
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
Submit	Submits the form.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomLink Class

Description
A DomLink represents all DOM elements that were specified using the <a> tag.

Inheritance Hierarchy

- *DomElement*
  - DomLink

Syntax
```
'Declaration
Public Class DomLink _
Inherits DomElement
```

Properties

Name	Description
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)

Name	Description
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
Select	Clicks a link.
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomListBox Class

Description
A DomListBox represents all DOM elements that were specified using the <select> tag.

Inheritance Hierarchy

- *DomElement*
  - DomListBox

Syntax
```
'Declaration
Public Class DomListBox _
Inherits DomElement
```

Properties

Name	Description
AllowsMultiSelect	whether the control is multiselectable.
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
ItemCount	the number of items in the listbox.
Items	the list of items in the listbox.
SelectedIndices	The indices of the selected item(s).
SelectedItems	The names of the selected item(s).
Text	The text of the control. (Inherited from *TestObject*)

Name	Description
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)

Name	Description
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)

Name	Description
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
MultiSelect	Selects a list of items.
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
Select	Selects an item.
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)

Name	Description
Unselect	Unselects an item.
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomRadioButton Class

Description
A DomRadioButton represents all DOM elements that were specified using <input type='radio'> tag.

Inheritance Hierarchy

- *DomElement*
  - DomRadioButton

Syntax

```
'Declaration
Public Class DomRadioButton _
Inherits DomElement
```

Properties

Name	Description
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
RadioListItemCount	The number of items in the radiolist. This property is not supported for mobile Web applications.
RadioListItems	the list of items in the radio list
RadioListSelectedIndex	The index of the selected item. Setter for this property was removed with Silk Test 17.5, as it was never functional. This property is not supported for mobile Web applications.
RadioListSelectedItem	The item that is selected in the radiolist. Setter for this property was removed with Silk Test 17.5, as it was never functional. This property is not supported for mobile Web applications.
Selected	whether the radio button is selected.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)

Name	Description
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)

Name	Description
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
RadioListSelect	Selects an item in the radio list.
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
Select	Selects the radio button.
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)

Name	Description
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomTable Class

Description
A DomTable represents the <table> tag. All methods and properties in this class are not supported for mobile Web applications.

Inheritance Hierarchy

- *DomElement*
  - DomTable

Syntax
```
'Declaration
Public Class DomTable _
Inherits DomElement
```

Properties

Name	Description
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)

Name	Description
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCell	Returns the specified cell.
GetCellText	Returns the content of a cell element.
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetColumnCount	Number of columns of the table.
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected

Name	Description
	by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetRow	Returns the DOM element for the specified row.
GetRowCount	Returns the number of rows
GetRows	Returns a list of rows (DomTableRows)
GetRowText	Returns the content of all cell elements of a specified row.
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider

Name	Description
	increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomTableRow Class

Description

A DomTableRow represents the <tr> tag. All methods and properties in this class are not supported for mobile Web applications.

Inheritance Hierarchy

- *DomElement*
  - DomTableRow

Syntax

```
'Declaration
Public Class DomTableRow _
Inherits DomElement
```

Properties

Name	Description
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant

Name	Description
	System.Drawing.Imaging.PixelFormat.Format32bppRgb can be used for this purpose. (Inherited from *TestObject*)
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetCell	Returns the DOM element in the given cell.
GetCellCount	Returns the number of cells in this table row
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)

Name	Description
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)

Name	Description
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown.

Name	Description
	The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

DomTextField Class

Description

A DomTextField represents DOM elements that were specified using one of the following tags: <input type='text'>, <input type='password'>, <input type='file'> or <textarea>.

Inheritance Hierarchy

- *DomElement*
  - DomTextField

Syntax

```
'Declaration
Public Class DomTextField _
Inherits DomElement
```

Properties

Name	Description
IsFocused	Whether the DOM element has focus. (Inherited from *DomElement*)
IsPassword	whether the text field is a password text field.
Text	The text of the control. (Inherited from *TestObject*)
Value	The value of the control, e.g.: text in a text control. (Inherited from *TestObject*)
Visible	Whether the object is visible. You can only locate visible objects, so by default the value is always *true*. However, you might need this property when an already located object in the application under test gets invisible during replay. (Inherited from *BrowserObject*)

Methods

Name	Description
CaptureBitmap	Saves a bitmap image of this object to a file. If you do not specify an absolute file name, the bitmap is saved to the default bitmap location on the machine that runs the Agent, which is %LOCALAPPDATA%/Silk/SilkTest/capturedBitmaps. The captured bitmap's pixels are stored in 32bit RGB order; when parsing the image the 4th byte (alpha channel) of every pixel must be ignored as it is used only for padding. In .NET, the constant *System.Drawing.Imaging.PixelFormat.Format32bppRgb* can be used for this purpose. (Inherited from *TestObject*)
ClearText	Clears the textfield's text
Click	Clicks on the object. (Inherited from *IClickable*)
DomClick	Invokes a click using the DOM API. Alternatively you can invoke the Click function which will use low level replay for DomElements and subclasses. (Inherited from *DomElement*)
DomDoubleClick	Invokes a double click using the DOM API. Alternatively you can invoke the DoubleClick function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DomMouseMove	Invokes a mouse move using the DOM API. Alternatively you can use MouseMove function which will use low level replay for DomElements and subclasses. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
DoubleClick	Double-clicks a mouse button on the object. (Inherited from *IClickable*)
ExecuteJavaScript	Evaluates the given JavaScript code within the parent document's context. (Inherited from *DomElement*)
Exists	Checks if any child object matching the locator exists for an object in the application under test. If the *timeout* parameter is passed the agent retries finding until the given timeout expires. If no timeout is specified and no object is found initially *false* is returned immediately. (Inherited from *TestObject*)
Exists	Checks if an object exists in the application under test. (Inherited from *TestObject*)
Find	Finds an object specified by an XPath locator. (Inherited from *TestObject*)
FindAll	Finds all objects specified by an XPath locator. (Inherited from *TestObject*)
GenerateLocator	Returns a locator for this object. (Inherited from *TestObject*)
GetChildren	Returns the child objects of this object. (Inherited from *TestObject*)
GetCssStyle	Retrieves the computed CSS style with the specified style name. (Inherited from *DomElement*)
GetDomAttribute	Gets the value of an object specific DOM attribute. (Inherited from *DomElement*)

Name	Description
GetDomAttributeList	Returns the names of all DOM attributes that are available for this element. This method is not supported for mobile Web applications. (Inherited from *DomElement*)
GetDynamicMethodList	Returns a list of methods (including their signature) that can be dynamically invoked on this TestObject. (Inherited from *TestObject*)
GetHtmlRect	Retrieves the rectangle of a DomElement relative to the BrowserWindow in HTML pixels. In contrast to the screen coordinates that are provided by the GetRect function, the coordinates provided by GetHtmlRect are not affected by the zoom level on mobile browsers. The click function also uses HTML pixel coordinates for all DomElements. We recommend to use GetHtmlRect for all position calculations within the browser. This will ensure that your scripts are cross-browser capable and your position calculations are not effected by changing zoom levels on mobile browsers. (Inherited from *DomElement*)
GetParent	Looks up the parent of this object in the test application. (Inherited from *TestObject*)
GetProperty	Returns the value of the specified property. (Inherited from *TestObject*)
GetPropertyList	Returns a list of property names that can be retrieved for the given object. (Inherited from *TestObject*)
GetRect	Returns the size and position of this object. Per default the returned coordinates are relative to the parent window that contains this object. (Inherited from *TestObject*)
GetText	Gets the visible text of a DOM element. (Inherited from *DomElement*)
Highlight	Highlights a DOM element. (Inherited from *DomElement*)
HighlightObject	Highlights this object. (Inherited from *TestObject*)
ImageClick	Clicks on specified image asset. (Inherited from *TestObject*)
ImageClickFile	Clicks on the specified image. (Inherited from *TestObject*)
ImageExists	Returns whether the specified image asset exists. (Inherited from *TestObject*)
ImageExistsFile	Returns whether the specified image exists. (Inherited from *TestObject*)
ImageRectangle	Returns the object-relative rectangle of the specified image asset. (Inherited from *TestObject*)
ImageRectangleFile	Returns the object-relative rectangle of the specified image. (Inherited from *TestObject*)
Invoke	Dynamically invokes a method on the test object. (Inherited from *TestObject*)
InvokeMethods	Dynamically invokes a sequence of methods starting at this TestObject. (Inherited from *TestObject*)

Name	Description
MouseMove	Moves the pointer to the specified location in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
PressKeys	Presses (but does not release) a set of keys or mouse buttons. (Inherited from *IKeyable*)
PressMouse	Presses (but does not release) a mouse button in the object. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ReleaseKeys	Releases a set of keys or mouse buttons. (Inherited from *IKeyable*)
ReleaseMouse	Releases a mouse button. This method is not supported for mobile Web applications. (Inherited from *IClickable*)
ScrollIntoView	Scrolls the DOM element into the visible area of the browser window. (Inherited from *DomElement*)
SetDomAttribute	Sets the value of an object specific DOM attribute. (Inherited from *DomElement*)
SetFocus	Gives focus to the control. (Inherited from *BrowserObject*)
SetProperty	Sets the value of the specified property. (Inherited from *TestObject*)
SetText	Sets the text.
TextCapture	Returns the text in this object's visible area. (Inherited from *TestObject*)
TextClick	Clicks in the center of the specified text. (Inherited from *TestObject*)
TextExists	Returns whether the specified text exists. (Inherited from *TestObject*)
TextRectangle	Returns the object-relative rectangle of the specified text. (Inherited from *TestObject*)
TypeKeys	Sends a set of keystrokes to the object. (Inherited from *IKeyable*)
TypePasswordKeys	Types an encrypted password into an object, for example a text field. (Inherited from *IKeyable*)
Unhighlight	Restores the original foreground and background colors. (Inherited from *DomElement*)
Verify	Executes a verification on the given asset in the context of this UI object. (Inherited from *TestObject*)
WaitForChildDisappearance	Waits until the child object specified by the 'locator' parameter does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForDisappearance	Waits until the object does not exist or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option

Name	Description
	OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)
WaitForObject	Waits for an object that matches the specified locator. If no object matches within an timeout an ObjectNotFoundException is thrown. The default timeout is 5 seconds and you can change the timeout by setting the value of the option *OPT_WAIT_RESOLVE_OBJDEF*. Use WaitForObject if the AUT takes a long time to display a specific object, for example when a transaction is processed before showing any results. By default, an action in the UI does not require a WaitForObject, because of the built-in synchronization. If you receive random timeout errors during normal script execution, consider increasing the default timeout instead of adding WaitForObject statements to your scripts. (Inherited from *TestObject*)
WaitForProperty	Waits until the property specified by the 'propertyName' parameter gets the value specified by the 'expectedValue' parameter or until the timeout is reached. If you don't explicitly pass a timeout using the optional timeout parameter, the timeout specified by the option OPT_WAIT_RESOLVE_OBJDEF will be used. An exception is thrown if the timeout is reached. (Inherited from *TestObject*)

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X