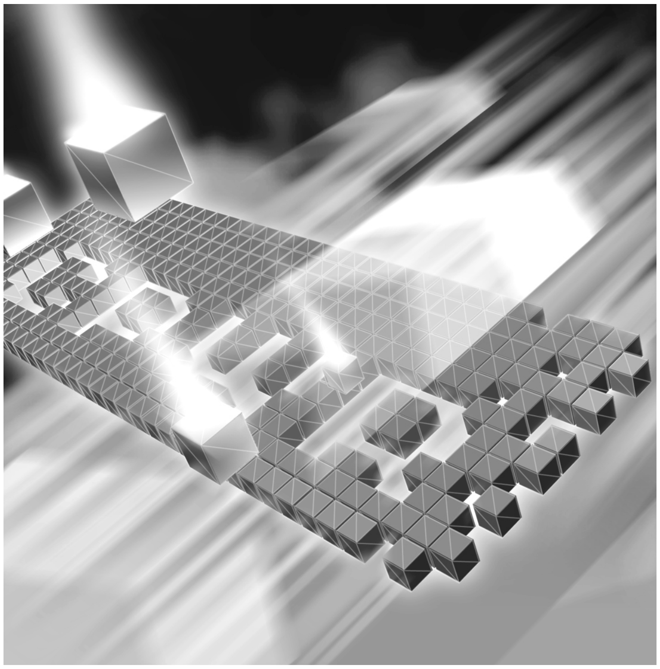


TestPartner

Getting Started Guide

Release 05.03

COMPUWARE®

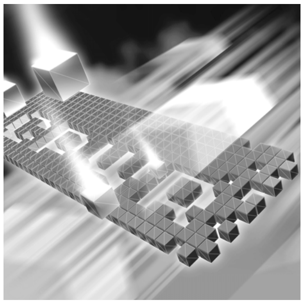
Customer support is available from our Customer Support Hotline or via our FrontLine Support Web site.

Customer Support Hotline:
1-800-538-7822

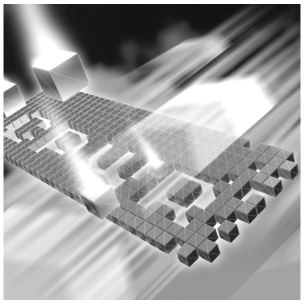
FrontLine Support Web Site:
http://frontline.compuware.com

Doc. CWMWGX53
January 7, 2005

Table of Contents



Introduction



How to Use This Guide

The TestPartner *Getting Started Guide* provides an overview of TestPartner and its basic functionality. It explains how to set up the system, develop scripts, define test conditions (checks and events), and how to view and interpret the results of testing. This guide provides general procedures to assist in getting started with the product. More comprehensive procedures are included in TestPartner's online help.

This manual is divided into the following chapters:

- Chapter 1: TestPartner Overview — Provides a brief product overview and describes the basic concepts of working with TestPartner.
- Chapter 2: Getting Started — Provides information on logging on to TestPartner, working with the TestPartner desktop and the Asset Browser, and using the Identify tool.
- Chapter 3: Database Maintenance Utility — Describes TestPartner's database update utility used to update database schema, unlock records, and compact databases.
- Chapter 4: Scripts — Describes the differences between using the Visual Navigator window and the VBA Code window for recording, editing, and playing back scripts.

- Chapter 5: Checks — Describes the various available checks and describes how to insert, edit, and verify the checks.
- Chapter 6: Events — Explains the creation, insertion, and editing of events.
- Chapter 7: Logs — Describes how to create and view TestPartner logs.

Who Should Read This Guide

The TestPartner *Getting Started Guide* is an introduction for TestPartner users. After completing the online tutorial, refer to this guide for explanations of the tasks and activities associated with preparing for, planning, building, and running test projects.

Familiarity with basic Microsoft Windows navigation is assumed. If this isn't the case, become familiar with the documentation accompanying Microsoft Windows before reading this guide.

Related Publications

In addition to the TestPartner *Getting Started Guide*, the TestPartner documentation includes the following other information sources:

- The TestPartner *Installation and Configuration Guide* provides system requirements and instructions for installing TestPartner. It also details basic setup and how to create and connect to a TestPartner database.
- The *Distributed License Management Installation Guide* provides instructions for licensing TestPartner. This guide is provided in PDF format.
- TestPartner*'s online help* provides information from the TestPartner User interface. If the **F1** key is pressed while using the software, context-sensitive online help is obtained. Online help can also be accessed by clicking **Help>Contents** from TestPartner's menu.

- TestPartner's *online language reference* details the commands available in TestPartner's Visual Basic for Applications (VBA) language extension, TPOSI. VBA is the language used in TestPartner scripts. The online language reference contains detailed information specific to command syntax, variants, operation, and script examples. It is intended for experienced TestPartner users who wish to utilize the scripting language to develop robust, sophisticated test procedures. The online language reference also integrates with the Microsoft *VBA Language Reference* and *Visual Basic Online Help* to provide information for all commands that can be used in TestPartner scripts, as well as Visual Basic functionality within TestPartner.
- TestPartner*'s tutorial* enables quick learning of the use of TestPartner. It introduces the basic steps of creating and working with an automated test script. The lessons step through the process of recording a script to test certain features of the sample target application, the NuBid auction website. Access the tutorial from TestPartner's **Welcome** dialog box or click **Help>Tutorial**.

Viewing and Printing Online Books

TestPartner's online books are provided in PDF format, Adobe Acrobat Reader 3.0 or above is needed in order to view them. To install the Adobe Acrobat Reader, click **Install Adobe Acrobat Reader** on the CD or go to Adobe's Web site at www.adobe.com.

Because PDF is based on PostScript, a PostScript printer is the most reliable way to print the online books. In most cases, PDF files can also be printed to PCL printers. If the PDF files cannot be printed, refer to Adobe's web site at www.adobe.com for troubleshooting information.

Online books can also be accessed from the **Start** menu. For example, the TestPartner *Getting Started Guide* can be accessed from Windows by clicking **Start>Programs>Compuware>TestPartner>Documentation**. The online books are also available from Compuware's FrontLine customer support Web site. For instructions on accessing FrontLine, see "FrontLine Support Web Site".

World Wide Web Information

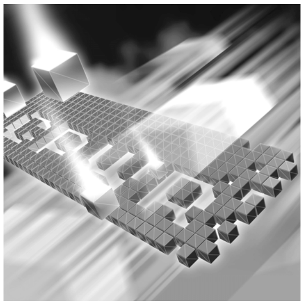
To access Compuware Corporation’s Web site, point the browser to http://www.compuware.com. The Compuware site provides a variety of product and support information.

FrontLine Support Web Site

Online customer support for Compuware products can be obtained via the FrontLine support Web site at http://frontline.compuware.com. FrontLine provides fast access to critical information about Compuware products. The site allows users to read or download documentation, browse frequently asked questions and product fixes, or e-mail questions or comments. The first time FrontLine is accessed, registration is required.

Chapter 1

TestPartner Overview



Advantages of Using TestPartner

TestPartner helps to plan, develop, and run test procedures and identify faults on application software and Web applications. With TestPartner, a user can record user sessions with applications, add validation functions to ensure that actual results match anticipated results, and replay the sessions later to ensure an application works as expected.

TestPartner 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, TestPartner's keystrokes and mouse actions are automatically generated by the software, rather than by hardware. TestPartner can determine test actions by monitoring the display to see how an application responds to inputs.

With automated testing in TestPartner, developed procedures can be run repeatedly, without error or any further manual input. This functionality reduces the amount of time spent testing to a fraction of the time manual testing takes. Automated tests can also be run overnight or on weekends, when machine time is easier to schedule. All of TestPartner's actions and all reactions from tested applications are recorded in a log, which can be reviewed at any time.

TestPartner provides a range of benefits for Windows-application and Web-application development organizations:

- Shorter development and testing cycles — Because TestPartner can run test suites unattended or at night, it can accelerate development and testing schedules, helping applications reach the market faster.
- Consistency — Because automated tests run the same way each time, TestPartner produces consistent, reproducible results.
- Increased productivity — Because TestPartner can run test suites without manual intervention, programmers and testing staff are freed for other tasks.
- Rapid test development — Create useful tests quickly and easily by navigating through an application. Set up templates to provide a standard framework for test scripts.

How TestPartner Works

TestPartner's method of test interaction with a target application is called *attaching*. When controlling a Windows application manually, select a window by clicking on it. Attaching to a window is comparable to clicking on the window to bring it into focus.

A Windows application displays various dialog boxes, each containing menus, edit boxes, lists, buttons, and other controls. By using the robust set of properties managed through *Object Maps*, TestPartner assigns each dialog box and control an attach name that uniquely identifies it. TestPartner then uses the attach name to locate the window and attach to it.

Similarly, a Web page may have many controls, such as buttons, to which TestPartner will also assign a unique attach name.

TestPartner allows complete control over how objects and their properties are recorded. For more information about the Object Map and customizing attach names, refer to TestPartner's online help.

The sections of this chapter introduce basic TestPartner testing concepts that should be understood before creating testing solutions with TestPartner, including:

- TestPartner projects
- Scripts
- Modules, shared modules, and class modules
- UserForms
- Object Maps
- Checks
- Events
- Logs

Each of the above items, except TestPartner projects, are defined and stored in TestPartner as a *test asset*. A test asset is any test element defined and stored in the database. Most test assets can be re-used or redefined without changing other test elements that refer to them. For example, if a check is created that is used in two different scripts, and the check is later modified, only the check's definition needs to be updated. All other assets that use the check (in this case, the two scripts) will automatically use the new information.

TestPartner Projects

A TestPartner project is a collection of test assets that includes assigned access privileges for each user on each project. Test assets can be viewed in the Asset Browser and include all of the standard Asset Browser objects, such as scripts, modules, checks, events, and logs.

A TestPartner project should not be confused with a VBA project. While TestPartner projects and VBA projects appear to share some of the same elements, such as UserForms and modules, VBA projects allow specific runtime properties and individual forms and procedures. In a TestPartner project, all code must be run in the context of a script.

Scripts

The actions that are used to test an application — making menu selections, typing information, clicking buttons — are all represented in TestPartner by commands that are automatically recorded in a script. See Chapter 4 for more information about scripts.

Modules, Shared Modules, and Class Modules

A module is a set of declarations designed to accomplish a particular task. In TestPartner, module assets are stored in the project in which they are created and must be included in any test script that requires them. In contrast, a TestPartner shared module is stored in a designated project and is automatically visible to all users in that project.

A class module is the definition of a class, including its properties and methods. Class module assets are not automatically visible to other assets in a project. Class modules can be used to explicitly control the scope of visibility.

Modules, shared modules, and class modules are described in more detail in TestPartner's online help.

UserForms

TestPartner UserForms are actually VBA UserForms. They are used to create windows and dialog boxes that can be displayed when scripts are run. UserForms are used to build a user interface to interact with test scripts. For example, a UserForm might be used to display a logon screen when the script is played back, so that can enter variable or confidential data can be inserted into a TestPartner script as it is running — this way, information does not need to be hard-coded into the script.

When a UserForm is created, the VBA Toolbox appears. The Toolbox allows standard Visual Basic controls, such as command buttons and tab strips, onto the form while it is being created. UserForms are described in more detail in TestPartner's online help.

Object Map

Object Map assets are TestPartner assets that assign an alias name to a screen object, such as a window or a control within a window. The alias represents the object's definition in the database, which includes all of the object's properties. After a screen object is registered as an Object Map asset, all references to it in scripts, checks, and events are made by its alias, rather than by its actual attach name. This way, if an object's properties change, only the Object Map asset needs to be changed.

Object Map assets provide two major benefits:

- They enable descriptive names to be substituted for complex actual attach names, which can make scripts easier to read.
- They eliminate a script's dependence on actual attach names, which may change if the target application is modified.

Maps can be created for objects (as described above) and images. Image Map entries identify a window by the image. Object Map entries identify a window by specific properties. In most cases, it is preferable to use object mapping. Image mapping allows one to capture a bitmap image, give it an alias, and store it as an Image Map entry in the Object Map. Like object mapping, image mapping provides a single point of maintenance. For example, if an icon to a button is changed, only the Image Map entry in the Object Map has to be changed. All scripts that refer to that button are automatically updated.

Although the Object Map is pre-defined to provide the most useful object types and definitions, the way that Object Map assets are created may be customized. Refer to TestPartner's online help for information about using the Object Map.

Checks

Checks verify that the target application is working correctly and providing the anticipated responses. Checks enable confirmation that the application is validating input, performing calculations correctly, saving data reliably, and reporting accurately. They compare actual responses to expected responses and reporting any discrepancies. See Chapter 5 for more information about checks.

Events

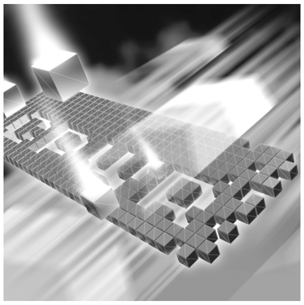
TestPartner can recognize unexpected conditions in the target application and respond appropriately if these conditions are defined in TestPartner as *events*. An event is a condition external to TestPartner, but internal to the computer system that can be tested by the script. Defining events enables a script to perform intelligently, allowing it to determine and define a script process flow that depends on external conditions. See Chapter 6 for more information on events.

Logs

Logs provide a detailed report on each command issued from the script. These logs allow users to quickly and easily analyze the outcome of the tests. While the script tests the target application or Web application, TestPartner writes all actions and the results to a log file. Each script line that involves the target application generates a log entry. If a check fails (that is, the target application did not respond to TestPartner's input as expected), both the expected and actual responses are stored in the log for comparison. The log can then display the detailed results of a specific TestPartner test script.

TestPartner's logging capability also enables the filtering of logs for quick sorting.

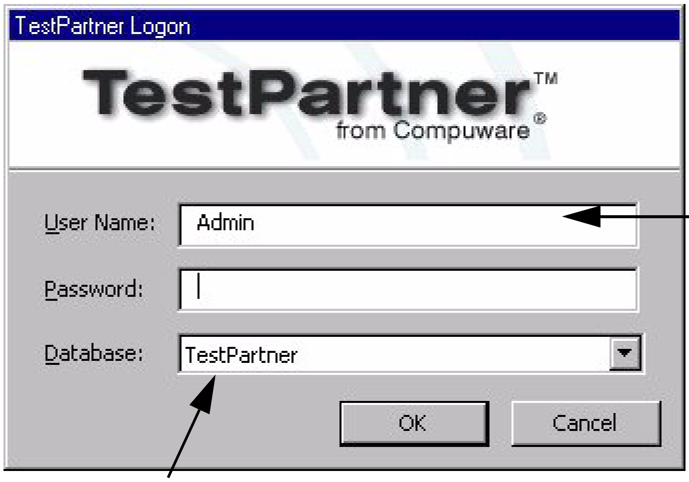
Chapter 2
Getting Started



Logging On and Starting TestPartner

Use the following procedure to launch TestPartner and log on to the TestPartner database.

1 Start TestPartner by clicking the taskbar's **Start** button, then choose **Programs>Compuware>TestPartner>TestPartner**. The TestPartner splash screen appears, followed by the TestPartner Logon dialog box.


TestPartner Logon
TestPartner™ from Compuware®
User Name:
Password:
Database:
OK
Cancel
Use the User Name and Password assigned by the TestPartner administrator
Use to switch databases

User Name: Admin

Password:

Database: TestPartner

2 Type a user name in the **User Name** field and type a password in the **Password** field.

   If using TestPartner 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 logon to prevent unauthorized access.

3 Select the TestPartner database to work with from the **Database** drop-down list. TestPartner's testing assets are stored in a test asset repository, or database.

   This multi-user repository offers centralized control of users and system access rights under password control. This allows for multiple database connectivity.

4 Click **OK** to log on. The TestPartner Welcome dialog box appears. This dialog box guides entry into the TestPartner desktop. It can be used to display scripts using the Visual Navigator window or the VBA Code window, to run the online tutorial, or to peruse the online documentation.

   This dialog appears after the first successful logon. TestPartner may be configured so that this dialog box does not appear during future access (see "Customizing TestPartner's Options" on page 26).

5 Choose **The Visual Navigator** or **The Code Window** option to open TestPartner's desktop. The Select Project dialog box appears. Because TestPartner automatically opens a new script each time it starts, and prompts to select a project. Refer to TestPartner's online help for information about administering users and adding projects.

6 Select an existing project from the drop-down list and click **OK**. TestPartner's desktop appears.

   If the **Code Window** option was selected, TestPartner's desktop appears, along with a new script window (open in the VBA Code window), the Output Window, the Workspace Window, and the Asset Browser (Choose **View>Asset Browser** if this is not in view). See "Understanding TestPartner's Desktop" on page 19 for information about working with TestPartner's desktop and windows.

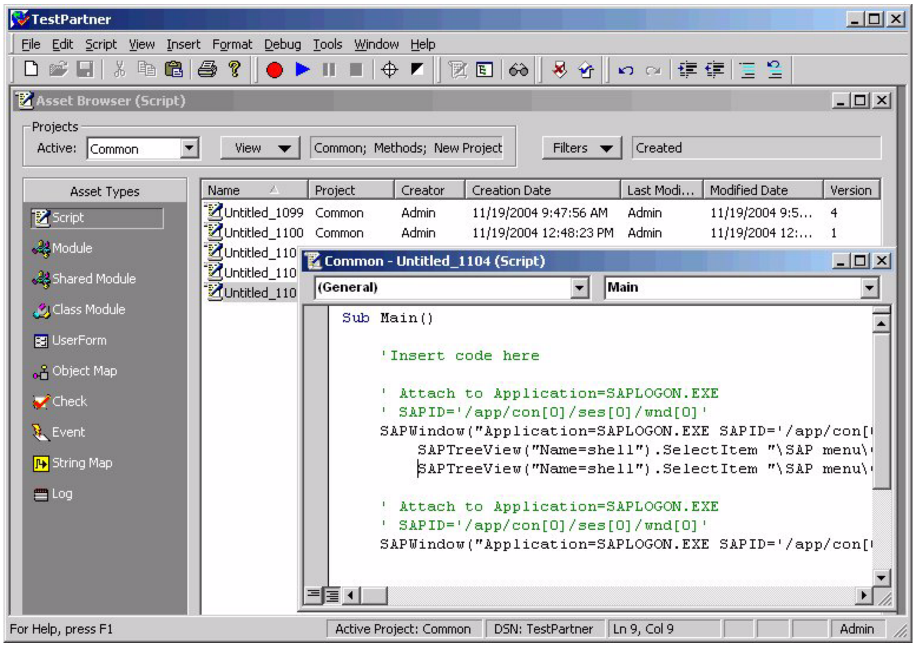
   If the **Visual Navigator** option was selected, TestPartner's desktop appears, along with a new script window (open in the Visual Navigator window) and the Asset Browser (Choose **View>Asset Browser** if this is not in view).

Understanding TestPartner’s Desktop

TestPartner’s desktop includes the TestPartner menus and toolbars, as well as TestPartner’s windows and the Asset Browser. With the exception of the menu bar, the user maintains control over all of the items that are displayed on the desktop.

The desktop also includes the status bar, the horizontal area at the bottom of the TestPartner desktop. The status bar provides information about the current state of what you are viewing in TestPartner, including the current active project and DSN for the currently accessed database, the Log on ID of the current user, and other contextual information. Also, When viewing a script in VBA code view, the current line and column number display in the status bar.

Figure 2-1. The TestPartner Desktop



Name	Project	Creator	Creation Date	Last Modi...	Modified Date	Version
Untitled_1099	Common	Admin	11/19/2004 9:47:56 AM	Admin	11/19/2004 9:5...	4
Untitled_1100	Common	Admin	11/19/2004 12:48:23 PM	Admin	11/19/2004 12:...	1
Untitled_110						
Untitled_110						
Untitled_110						

```
Sub Main()

    'Insert code here

    ' Attach to Application=SAPLOGON.EXE
    ' SAPID='/app/con[0]/ses[0]/wnd[0]'
    SAPWindow("Application=SAPLOGON.EXE SAPID='/app/con[
        SAPTreeView("Name=shell").SelectItem "\SAP menu\
        SAPTreeView("Name=shell").SelectItem "\SAP menu\

    ' Attach to Application=SAPLOGON.EXE
    ' SAPID='/app/con[0]/ses[0]/wnd[0]'
    SAPWindow("Application=SAPLOGON.EXE SAPID='/app/con[
```

- Use the **Asset Browser** to work with TestPartner assets such as Scripts and Checks.
- View data in various windows, such as the Output window, which displays compile errors, script errors, and search results. See TestPartner online help for more information about other windows in the desktop.
- Once any desktop window is closed, open it again by clicking **View>Item** from the menu bar.

- If an open item loses focus, but is still open, it can be brought into focus again by clicking **Window>Item** from the menu bar.

The desktop's menus, toolbars, and the Asset Browser are described in this chapter. Other desktop windows are described in TestPartner's online help.

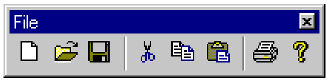
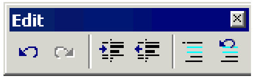
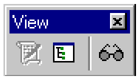
Menus and Toolbars

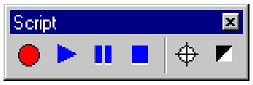
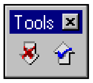
TestPartner has two types of menus: built-in and shortcut. The TestPartner menu bar appears at the top of the TestPartner desktop. Many of the TestPartner menus or menu commands apply only when editing or debugging scripts. If this is the case, the menu is enabled, but some commands will be disabled.

Shortcut menus contain frequently used commands that appear when TestPartner is right-clicked or SHIFT+F10 is pressed. The items on a shortcut menu depend on the active window and recently performed actions.

TestPartner's toolbars allow quick access to commands. Use toolbars to create new assets, start and stop recording, and undo and redo changes. The TestPartner desktop typically displays several small, modular toolbars. Each toolbar can be opened, closed, or moved to any location on the desktop. If the mouse pointer is positioned over the toolbar button, a tooltip appears with a brief description of the button's function. Table 2-1 provides a quick reference to toolbar functions.

Table 2-1. TestPartner's Toolbars.

Toolbars	Usage
	**File Toolbar:** Contains general Microsoft buttons for creating, opening, and saving TestPartner assets.
	**Edit Toolbar:** Contains buttons to undo and redo, indent and outdent, or block comment lines of code while working with scripts in VBA code view.
	**View Toolbar:** Contains buttons that allow toggling between the VBA Code window and the Visual Navigator window. The Asset Browser can also be accessed.

Toolbars	Usage
	**Script Toolbar:** Contains buttons that allow control of the script recording process. Scripts can be recorded, played back, paused, and stopped. This toolbar also contains a button that accesses the Identify and Compare Attach Data tools.
	**Tools Toolbar:** Contains buttons that allow TrackRecord defect submission.

Using the Asset Browser

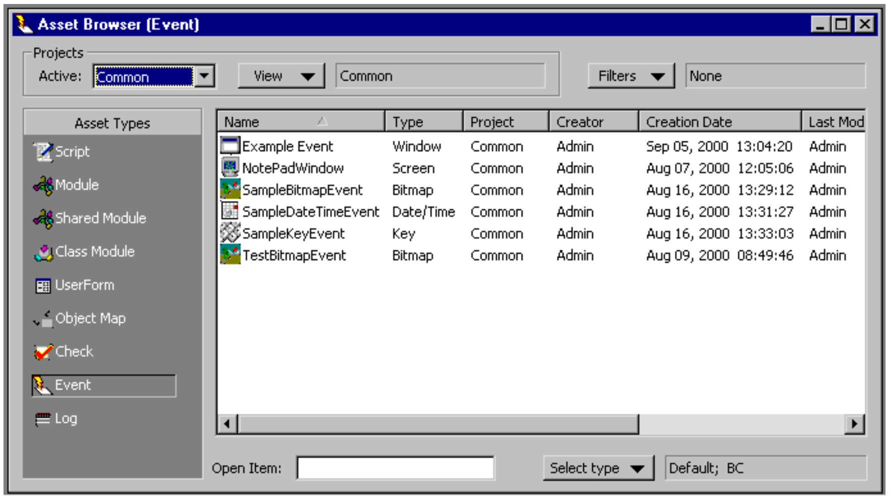
The Asset Browser is an important aspect of the TestPartner desktop. This is the window that is used most frequently as scripts and test assets are built. The Asset Browser provides a single point for creating, managing, and viewing all types of test assets. The Asset Browser contains the following items:

- Scripts
- Modules
- Shared modules
- Class modules
- UserForms
- Object Maps
- Checks
- Events
- Logs

Each of these items can be defined and stored in TestPartner as a *test asset*. A test asset is any test element that is defined and stored in the database. Most test assets can be re-used or redefined without requiring a change to the other test elements that refer to them. For example, if a check is created that is used within two different scripts, and the check is later modified, only the check's definition needs to be updated. All other assets that use the check (in this case, the two scripts) will automatically use the new information — there is no need to edit both the scripts and the checks.

Likewise, use the Asset Browser to create a check that is very similar to a previously defined check. Rather than create a new check from scratch, save time by duplicating the check from the Asset Browser and changing the copy. Figure 2-2 shows the Asset Browser displaying Event assets.

Figure 2-2. Overview of the Asset Browser



Name	Type	Project	Creator	Creation Date	Last Mod
Example Event	Window	Common	Admin	Sep 05, 2000 13:04:20	Admin
NotePadWindow	Screen	Common	Admin	Aug 07, 2000 12:05:06	Admin
SampleBitmapEvent	Bitmap	Common	Admin	Aug 16, 2000 13:29:12	Admin
SampleDateTimeEvent	Date/Time	Common	Admin	Aug 16, 2000 13:31:27	Admin
SampleKeyEvent	Key	Common	Admin	Aug 16, 2000 13:33:03	Admin
TestBitmapEvent	Bitmap	Common	Admin	Aug 09, 2000 08:49:46	Admin

Table 2-2. Main Controls in the Asset Browser

Control	Description
Active Project List	Set the active project
Asset Types List	Select what types of assets to view
View List	Select what projects to view
Open Item Field	Type the name of an asset to open
Filter List	Filter assets based on set criteria

Changing and Viewing Projects

The Asset Browser displays all test assets that belong to the projects that are selected in the View dialog box. A TestPartner project is a collection of test assets with assigned access privileges. Only the test assets in projects for which the current used has been assigned access can be viewed.

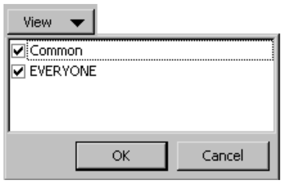
By default, new assets are always added to the active project and are available only in that project. The Common project is used to store assets that are used in multiple projects, because assets that are in the Common project are available for all other projects. Each asset (whether from the Common project or from another project) must be explicitly included in the TestPartner script that uses it.

The active project can be changed by selecting it from the Active list in the Projects area of the window. To change the active project or to view additional projects:

1 Click **View>Asset Browser**. The Asset Browser appears.

2 To change the active project, select the desired project from the **Active** list in the **Project** area.

3 To add a project to the Asset Browser view, click the **View** button. The **View** dialog box appears. This dialog box displays the list of projects to which the current user has access.



4 Select the projects to view and click **OK**. The view of the Asset Browser changes to reflect the test assets contained in all of the selected projects.

Customizing Grid Columns and Sort Order

Any asset's grid can be rearranged, sorted by order, or sorted by direction:

- **Column Headings:** Change the order in which the column headings appear by dragging and dropping.
- **Sort Order:** By default, the list is sorted in ascending order. To change the sort order, click the column heading.

Viewing Assets

To view existing assets, click the appropriate button in the **Asset Type** area of the **Asset Browser** (see Figure 2-2 on page 22). TestPartner displays a corresponding list of the assets in the right pane grid of the Asset Browser. This view can also be used to rename, copy, delete, and view asset properties by right-clicking the asset name and choosing the appropriate command.

Adding Assets

Although most test assets are created when recording a script or by inserting them into an existing script, they can also be created by adding them directly to the list of assets in the Asset Browser.

To create a new asset in the Asset Browser:

1 Click **View>Asset Browser**. The Asset Browser appears.

2 Click the appropriate button in the **Asset Types** pane. The Asset Browser displays a list of existing assets in the right pane.

3 Right-click the **Asset** button to create and choose **New <*Asset Type*>**.

Note: Select the **New** command from the shortcut menu to select an asset type to create.

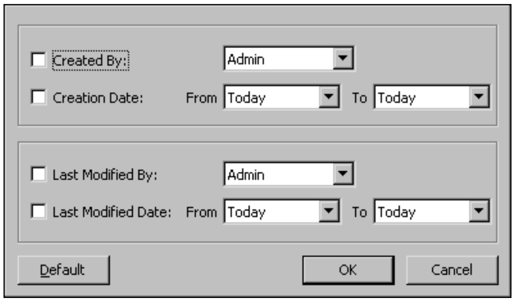
New assets are added to the active project and are available only in that project (see "Changing and Viewing Projects" on page 22 for information on changing the active project).

Each asset that is created in the Asset Browser (whether from the Common project or from the project containing the script) must be added to a TestPartner script.

Filtering Assets

Filters create a view of only those records that are of interest. Use the following procedure to filter the view of the Asset Browser:

1 Click the **Filters** button in the Asset Browser window. The Filters dialog box appears.



- [ ] Created By: Admin
- [ ] Creation Date: From Today To Today
- [ ] Last Modified By: Admin
- [ ] Last Modified Date: From Today To Today

Specify the filter criteria. If more than one filter is selected, the asset must meet all of the criteria to be displayed.

◇ If **Created By** is clicked, select the name of a user who created the assets.

◇ If **Creation Date** is clicked, specify the start and end date to see all assets created during that period.

- If **Last Modified By** is clicked, select the name of a user who modified the asset.
- If **Last Modified Date** is clicked, specify the start and end date to see all assets that were modified during that period.

2. Click **OK** to apply the filter. TestPartner applies the filter to the list of asset items currently displayed. To clear the filter and display all the items again, click the **Filters** button and then click the **Default** button on the filters dialog box.

Working with Asset Versions

TestPartner automatically stores versions of the following asset types:

- Scripts
- Modules
- Shared Modules
- Class Modules
- Checks
- Events

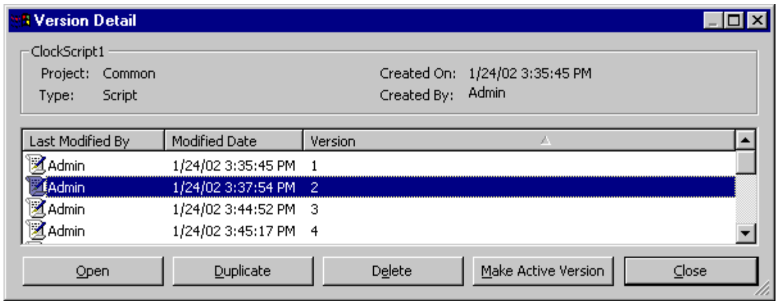
Each time an asset is modified, a new version is stored in the database. A list of all the versions and version information for an asset can be viewed, including the user that last modified the asset, the date and time of the modification, and the version number.

Use asset versions to restore an older version, making it the current asset. This is useful if an asset's definition was modified and an older version is needed. The Version Details dialog box can be used to edit, copy, or delete a specific version of the asset.

To view all versions of an asset:

1. Click **View>Asset Browser**.
2. Click the asset type button in the **Asset Types** pane. The Asset Browser displays a list of existing assets in the right pane.

3 Select the appropriate asset and click **File>Show All Versions**. The Version Detail dialog box appears.



Project: Common
Type: Script
Created On: 1/24/02 3:35:45 PM
Created By: Admin

Last Modified By	Modified Date	Version
Admin	1/24/02 3:35:45 PM	1
Admin	1/24/02 3:37:54 PM	2
Admin	1/24/02 3:44:52 PM	3
Admin	1/24/02 3:45:17 PM	4

4 Use the buttons across the bottom of the Version Detail dialog box to open, copy, and delete versions of the asset. Make any version the currently active version by selecting it and clicking the **Make Active Version** button.

Customizing TestPartner's Options

TestPartner's options control the way TestPartner functions when starting the application and when recording and playing back scripts. For example, TestPartner's options might be changed if several test scripts in a row are to be run, and the preference is to view the logs after all of the tests have completed, rather than after each test completes.

Change any options using TestPartner's Options dialog box. To modify options:

1 Click **Tools>Options**. The Options dialog box appears. The right pane of the dialog box displays the available General options.

2 Turn an option on or off by selecting the option and choosing **Yes** or **No** from the drop-down list. If the option requires a specific value, select the option and edit the corresponding value.

3 Click **OK** to save settings and close the Options dialog box.

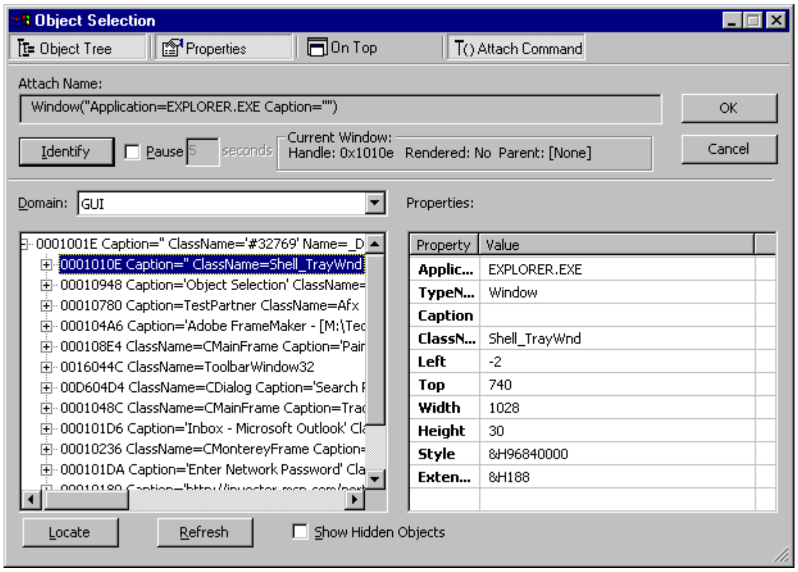
Using the Identify Utility

The Identify utility supplies information about the windows displayed on your screen. It is used within several TestPartner modules. For example, checks and events use it when identifying the target window.

Even though Identify is integrated into the creation of many TestPartner assets, it can also run manually from the desktop. Use Identity to find out what is on the screen, to investigate problems with scripts, or to generate attach names. It can also be used in conjunction with ComSpy to determine the attach name of running COM objects with exposed interfaces (refer to TestPartner's online help).

To use Identify:

1 Click the **Identify** button on the toolbar. TestPartner minimizes and the Object Selection dialog box appears.



Attach Name:
Window("Application=EXPLORER.EXE Caption="")

☐ Pause 5 seconds

Current Window:
Handle: 0x1010e Rendered: No Parent: [None]

Domain: GUI

```
0001001E Caption='' ClassName='#32769' Name=_D
   0001010E Caption='' ClassName=Shell_TrayWnd
   00010948 Caption='Object Selection' ClassName=
   00010780 Caption=TestPartner ClassName=Afx
   000104A6 Caption='Adobe FrameMaker - [M:\Tec
   000108E4 ClassName=CMainFrame Caption='Pair
   0016044C ClassName=ToolbarWindow32
   00D604D4 ClassName=CDialog Caption='Search F
   0001048C ClassName=CMainFrame Caption=Trac
   000101D6 Caption='Inbox - Microsoft Outlook' Cla
   00010236 ClassName=CMontereyFrame Caption=
   000101DA Caption='Enter Network Password' Cla
```

Properties:

Property	Value
Applic...	EXPLORER.EXE
TypeN...	Window
Caption	
ClassN...	Shell_TrayWnd
Left	-2
Top	740
Width	1028
Height	30
Style	&H96840000
Exten...	&H188

☐ Show Hidden Objects

Initially, the Object Selection dialog box captures details for all available objects. The Identify dialog box then displays each object's properties as it is selected in the tree view.

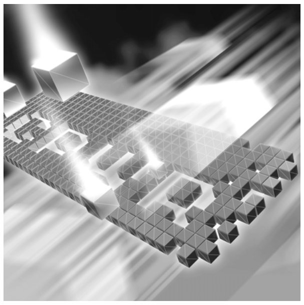
To see the properties of a specific control, use the Object Selection dialog box's **Identify** button.

2 Click the **Identify** button on the Object Selection dialog box. The mouse pointer changes to a cross-hair. Move the mouse pointer over the object of interest. As each window is identified, it is highlighted with a shaded border and a yellow ToolTip appears that displays the object type.

3 Click the object while the ToolTip is still present, the item will be automatically highlighted in the object tree and its property details will be displayed in the Properties pane.

Note: To see the details of a window that requires a mouse action to activate (such as a context menu or a system menu), select the **Pause** checkbox, then enter the number of seconds TestPartner should delay before recording the object under the cross-hair pointer. See TestPartner online help for more details about using the Identify utility.

Chapter 3
Database Maintenance Utility



Starting the Database Maintenance Utility and Opening Databases

TPMAINT.EXE is TestPartner's database update utility. This utility performs several maintenance tasks, such as updating the database schema, unlocking records in the database, and compacting the database.

Note: The Database Maintenance utility is used to populate the database instance with the appropriate TestPartner tables and data. Before the Database Maintenance utility can be used to create a new TestPartner ODBC database, a System Administrator must create a new ODBC data source. For more information, refer to the TestPartner Installation and Configuration Guide.

Use the following procedure to start the Database Maintenance utility:

1 Click **Start>Programs>Compuware>TestPartner>Utilities>Database Maintenance**. The Database Maintenance utility main window appears.

2 Click **File>Open Database** and choose the type of database from the submenu (for example, Access, Oracle, or SQL Server). Depending on the type of database selected, a dialog box similar to the following appears.

3 Enter the database name, user ID (must correspond to a database user), and password for the TestPartner database instance, if necessary (as with Oracle and SQL Server).

Note: For SQL Server, only SQL Server ID authentication is supported. SQL Server Authentication Mode must be set to: SQL Server and Windows (Mixed Mode). Please refer to the TestPartner Installation and Configuration Guide for more details.

4 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, TestPartner displays a message. See "Performing Maintenance on a Database with Connected Users" on page 30 for more information.

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 the **Send** button.

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.

Creating New Databases

The Database Maintenance utility enables creation of new databases to use as the TestPartner asset repository. A new Microsoft Access database can be created as well as a new SQL Server database or a new Oracle database. The proper authority is required to create or upgrade a database.

Note: If a new database is created that uses the same name and location as an existing database, the Database Maintenance utility will **overwrite** any TestPartner information already contained in the specified database to create a new, empty database.

Creating New Microsoft Access Databases

Use the following procedure to create a new Microsoft Access database:

1 Start the Database Maintenance utility (see "Starting the Database Maintenance Utility and Opening Databases" on page 29).

2 Click **File>New Database>Access**. The Access Database Connection dialog box appears.

3 Type the Data Source Name (DSN) for the Access database or click the **Browse** button to select an existing database.

4 Enter the file name for the Access database, or click the **Browse** button to select an existing database.

5 Click the **Create** button. If an existing database is entered, a message appears, asking if the desire is to overwrite.

The Create Data Source dialog box appears and displays the progress of the database creation.

Creating New Oracle Databases

To create a new Oracle database:

1 Start the Database Maintenance utility (see "Starting the Database Maintenance Utility and Opening Databases" on page 29 if necessary).

2 Click **File>New Database>Oracle**. The Oracle Data Source Connection dialog box appears.

3 Type the name of the new Oracle data source or click the **Browse** button to select an existing data source. Type aUser ID, Password, and a brief description of the DSN (the **Schema** field is automatically filled in with the User ID). Click the **Create** button.

Required: The User ID must have RESOURCE privileges. If the database instance is Oracle 9 or higher, the User ID must also have SELECT_CATALOG_ROLE privileges.

Note: A table called TP_DSNSCHEMA must exist in the SYSTEM schema. If the table does not already exist in the SYSTEM schema, a System Login dialog box appears, requiring the password for SYSTEM. Type the password, and click **OK**. The table is created.

Only one schema can be associated with a particular DSN name (ODBC connection). If the chosen DSN name is already associated with a schema, a DSN/Alias Already Exists dialog box appears.

4 Click the **Replace** button to change the association of the DSN name from its current schema to the new schema (users can no longer access TestPartner tables in the old schema), or click the **Alias** button to create an alias for the DSN name. A DSN Alias Creation dialog box appears.

5 Type an alias name for the DSN. This alias will be associated with the current schema. Click **OK** (see "View DSN Schema Associations" on page 33).

The Creating Data Source dialog box appears and displays the progress of the database creation.

Creating New SQL Server Databases

To create a new SQL Server Database:

1 Start the Database Maintenance utility (see "Starting the Database Maintenance Utility and Opening Databases" on page 29).

2 Click **File>New Database>SQL Server**. The SQL Server Data Source Connection dialog box appears.

3 Type the name of the new SQL Server data source or click the **Browse** button to select a data source name.

4 Type the name of the database Owner, User ID, and Password. Click the **Create** button.

Note: The User must have been previously created. Use the SQL Server Enterprise Manager to create users.

Only one schema can be associated with a DSN name (ODBC connection). If the chosen DSN name is already associated with a schema, a DSN/Alias Already Exists dialog box appears.

5 Click the **Replace** button to change the association of the DSN name from its current schema to the new schema (users will no longer be able to access TestPartner tables in the old schema), or click the **Alias** button to create an alias for the DSN name. A DSN Alias Creation dialog box appears.

6 Type an alias name for the DSN, this alias will be associated with the current schema. Click **OK** (see "View DSN Schema Associations" on page 33).

  The Creating Data Source dialog box appears and displays the progress of the database creation.

View DSN Schema Associations

To view which Schemas are associated with which DSN names:

Note: Schema/DSN relationships only apply to Oracle or SQL Server databases. They do not apply to Access databases.

1 If a database is not already open, click **File>Open Database** and select either an Oracle or SQL Server database.

2 Once the database is open, click **View>DSN Schema Associations**.

Updating Database Versions

When TestPartner is upgraded to the current version, the existing database must be converted to be compatible with the new software. TestPartner 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 TestPartner schema. The database can be converted with the Database Maintenance utility.

To convert a database:

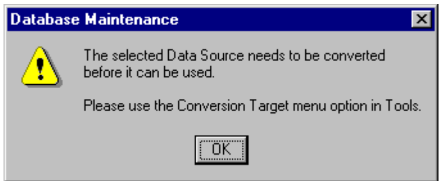
1 If necessary, cancel the TestPartner logon request and ensure that users are not logged on to the database.

2 Start the Database Maintenance utility and open the database (see "Starting the Database Maintenance Utility and Opening Databases" on page 29). The Database Maintenance utility main window appears.

3 Click **File>Open Database** and choose the type of database from the cascading menu.

4 The Database Connection dialog box appears. Type the database name and password for the database or click the **Browser** button to select a database to convert.

5 Click **Open**. If the database requires updating, the following message displays:


Database Maintenance

The selected Data Source needs to be converted before it can be used.

Please use the Conversion Target menu option in Tools.

OK

6 Click **OK**. The database opens and the Database Maintenance utility's Tools menu options are enabled.

7 Click the **Convert Database** toolbar button. A dialog box asks if the current database should be backed up prior to conversion. Choose to back up the database. The Convert Database Dialog box appears.

8 Click the **Convert Now** button to update a database to the current schema version. The Progress area gives a visual indication of the progress of the conversion.

The conversion speed is related to the amount of data being converted. Databases with large amounts of data may experience long conversion times. Conversion from a database used in TestPartner 05.02.01 or earlier to TestPartner 05.03 may take longer to convert data types and character strings to Unicode character strings. After the conversion, the current database version number appears.

9 Click **Close** when the conversion completes.

Copying to Other TestPartner Databases

TestPartner's Database Maintenance utility can copy from one database type to another, or to make a duplicate of another database. For example, if a site uses Microsoft Access as the TestPartner asset repository, and the desire is to store a TestPartner data in an Oracle database, use the Database Maintenance utility to copy the TestPartner data stored in the Access database to the Oracle database. When the copy utility is used, all TestPartner data is automatically copied to a format that is readable by the target database.

To copy from one database type to another:

Note: 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 (see "Starting the Database Maintenance Utility and Opening Databases" on page 29).
3. Click the **Copy** toolbar button. The Copy Destinations dialog box appears.
4. Select the type of database to copy to and click **OK**.

Copying To an Access Database

If copying to a Microsoft Access database, the Access Database Connection dialog box appears.

1. In the Access File field, type a name for the Access database or click the **Browse** button to select an existing database.
2. Click the **Copy** button. The Copy Table Status dialog box appears and displays the progress of the database copy operation.

Copying to a SQL Server Database

If copying to a SQL Server database, the SQL Server Data Destination Connection dialog box appears.

1. Type the name of the SQL Server data source or click the **Browse** button to select a data source name.

Note: The owner name defaults to `dbo`.

2. Type the user ID. The user must have dbo privileges.
3. Type the password for the TestPartner database instance on the database server. This User ID and password is used to attempt a connection to the TestPartner database instance on the database server.
4. Click the **Copy** button.

If the owner name, user ID, and password are accepted by the database server, the Data Source Creation Status dialog box appears and displays the progress of the database copy operation.

Copying to an Oracle Database

If copying to an Oracle database, the Oracle Data Destination Connection dialog box appears.

1 Type the name of the new Oracle data source or click the **Browse** button to select an existing data source name.

Note: The Schema and User ID fields 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 the **Copy** button. If the schema name, user ID, and password are accepted by the database server, the Data Source Creation Status dialog box appears and displays the progress of the copy operation.

The Database Maintenance utility begins populating the TestPartner database instance with the necessary TestPartner tables.

Unlocking Database Records

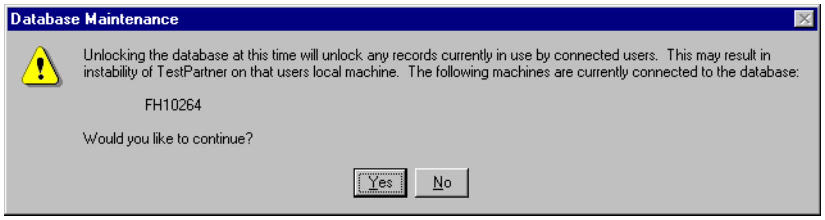
When a script, check, or any other TestPartner resource for editing is open, the record is locked to prevent other users from editing the same resource simultaneously. When saving the resource or closing the system, the locks are removed. If you experience a system crash while using TestPartner, it is possible that records that were in use will remain locked on start-up. TestPartner will attempt to unlock records that were inadvertently locked, and originally intended to remain unlocked, after a system crash.

Use the Database Maintenance utility to unlock all locked records. Ensure that all users are logged off the system before using the program. If there are users connected, a message will appear.

Use the following procedure to unlock all locked records:

1 Start the Database Maintenance utility and open the database (see "Starting the Database Maintenance Utility and Opening Databases" on page 29).

2 Click **Unlock Records**. If there are users connected to the database a message similar to the following appears:


Database Maintenance

Unlocking the database at this time will unlock any records currently in use by connected users. This may result in instability of TestPartner on that users local machine. The following machines are currently connected to the database:

FH10264

Would you like to continue?

3 Click **Yes** to unlock all locked records. A message box will verify that the database has been unlocked.

4 Click **OK.**

Compacting a Microsoft Access Database

As a general rule, it is a good idea to perform regular maintenance on any frequently used database. To ensure optimum performance, regularly compact the database. For example, if the database is used on a daily basis, perform weekly database maintenance.

Over time, as resources are updated or deleted, the database file becomes fragmented. Fragmentation wastes disk space and can impair performance. The Database Maintenance program allows you to compact a Microsoft Access database, removing unwanted records and rewriting the data to a contiguous area of the disk.

Note: Database Compact operations should not be performed on network databases. Instead, copy the database should to a local workstation, and then copy it back to the network location. This ensures that network disruptions and interruptions will not interfere with the database maintenance task.

To compact a Microsoft Access database:

1 Ensure that all users are logged off the system and close TestPartner.

2 Start the Database Maintenance utility and open the database (see "Starting the Database Maintenance Utility and Opening Databases" on page 29).

3 Click **Compact**. A confirmation dialog box appears.

4 Click **Yes** to proceed. When complete, a dialog box appears that indicates that the database has been successfully compacted.

5 Click **OK**. The database is now available for use.

Note: An Access database will not register a decrease in file size when assets are deleted from it until it is compacted using this procedure.

Configuring a Database

The Database Maintenance utility enables you to configure settings that are used during compact, convert, and copy operations.

When a database is compacted or converted, the current version can be backed up. Use the Configuration 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, and you convert or compact again, 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.

To configure a database:

1. Ensure that all users are logged off the system and close TestPartner.
2. Start the Database Maintenance utility and open the database (see "Starting the Database Maintenance Utility and Opening Databases" on page 29).
3. From the **Options** menu, choose **Configure**. The Configure Operations for Maintenance System dialog box appears.
4. Double click the options to set the number of backups to retain from compacts and conversions, and to set the number of records to commit.

Running an Integrity Check

Use the Database Maintenance utility to verify the integrity of a database, and to correct errors it may encounter.

To run an integrity check:

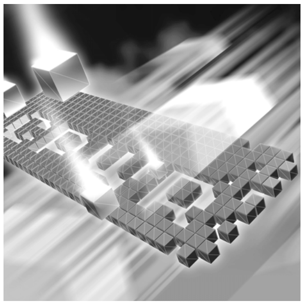
1. Open the database, and click **Tools>Run Integrity Check**. TestPartner checks the database for known errors.

   If an error is encountered, a message dialog box appears.

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 directory where TestPartner is installed, for example, **C:\Program Files\Compuware\TestPartner\IntegrityCheck.log**.

Chapter 4
Scripts



Using Scripts

TestPartner uses scripts 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. After a script is recorded, TestPartner generates all of the keystrokes and mouse clicks when the script is played back.

The actions taken to test an application — making menu selections, typing data, checking the way it is processed, and so on — are represented in TestPartner scripts' VBA commands. These commands are inserted in the script and can be modified and played back.

Modify scripts to include "hand-coded" statements that cannot be recorded, to make amendments to reflect changes in the target application or Web page, or to create new scripts by cutting and pasting code from old scripts.

The TestPartner online help demonstrates in detail how to save, close, edit, copy, and rename scripts.

This chapter explains how to use the basic script functions. It guides you through the essential processes of creating, saving, opening, and deleting scripts — as well as several other tasks. It does not explain the language used within the scripts; refer to the VBA online help provided with TestPartner for specific information about these commands.

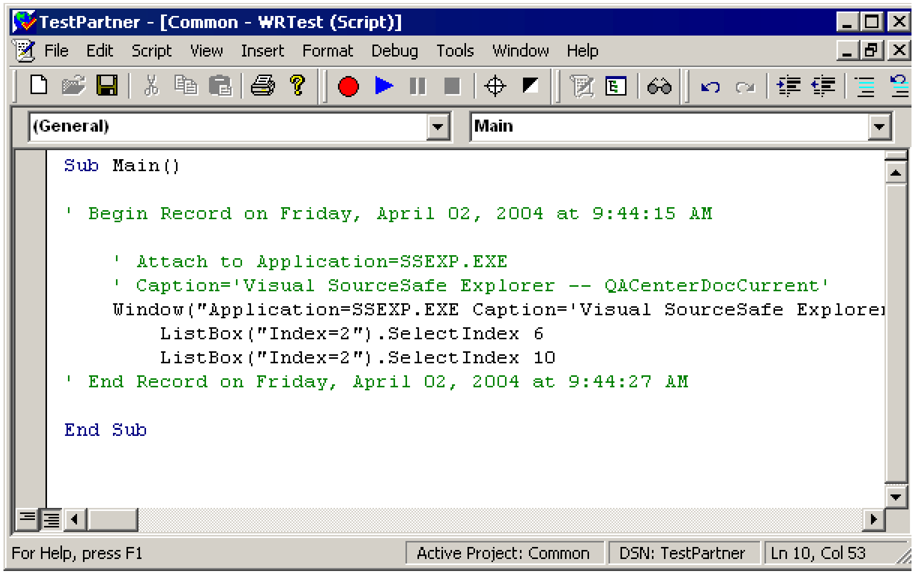
Recording Scripts

TestPartner uses VBA as its scripting language. TestPartner provides a flexible scripting environment in which scripts can be recorded using either the VBA Code window or the Visual Navigator window. When a script is recorded from the VBA Code window, TestPartner writes and displays the script in VBA, so code that is being generated can be seen as the recording session takes place. When a script is recorded, everything done in the Windows environment will be recorded until the session is stopped. In contrast, when a script is recorded using the Visual Navigator window, TestPartner creates and displays the script in an outline, or tree view, using nodes to represent each action or event that occur.

Record a script from the VBA Code window or from the Visual Navigator Window. The content of the script will be the same, regardless of which method is chosen — only the presentation of the script's content is different. In fact, the same script can be viewed or played back using either window.

Choosing the VBA Code Window to Record Scripts

The VBA Code window presents TestPartner scripts in a text-based code view (see Figure 4-1). Viewing scripts in the VBA Code window enable extensive script edits, placing the full power of VBA at your disposal.


TestPartner - [Common - WRTest (Script)]
File Edit Script View Insert Format Debug Tools Window Help
(General)
Main

```
Sub Main()

' Begin Record on Friday, April 02, 2004 at 9:44:15 AM

    ' Attach to Application=SSEXP.EXE
    ' Caption='Visual SourceSafe Explorer -- QACenterDocCurrent'
    Window("Application=SSEXP.EXE Caption='Visual SourceSafe Explorer
        ListBox("Index=2").SelectIndex 6
        ListBox("Index=2").SelectIndex 10
' End Record on Friday, April 02, 2004 at 9:44:27 AM

End Sub
```

```
For Help, press F1    Active Project: Common    DSN: TestPartner    Ln 10, Col 53
```

Figure 4-1. Script Viewed in the VBA Code Window

Use one of the following methods to record or view scripts in the VBA Code window:

- Choose the **Code Window** option from TestPartner's Welcome dialog box. TestPartner opens a new untitled script in the VBA Code window.

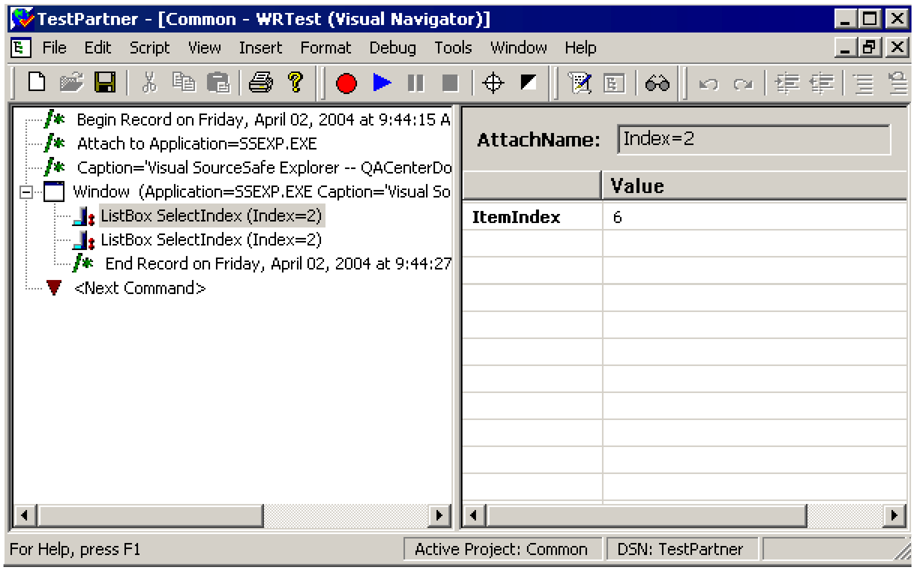
  The Welcome dialog box is an optional display that can be turned on or off using TestPartner's General options (see "Customizing TestPartner's Options" on page 26).

- With TestPartner already open, click the **Show Script Code** Window button. When selected, TestPartner opens a new untitled script in the VBA Code window.

Choosing the Visual Navigator Window to Record Scripts

The Visual Navigator window presents TestPartner scripts in an outline view (see Figure 4-2). Scripts viewed in the Visual Navigator window are easy to understand and do not require experience with the VBA language.

Figure 4-2. Script Viewed in the Visual Navigator Window


TestPartner - [Common - WRTest (Visual Navigator)]
File Edit Script View Insert Format Debug Tools Window Help
For Help, press F1
Active Project: Common
DSN: TestPartner

Begin Record on Friday, April 02, 2004 at 9:44:15 A
Attach to Application=SSEXP.EXE
Caption='Visual SourceSafe Explorer -- QACenterDo
Window (Application=SSEXP.EXE Caption='Visual So
ListBox SelectIndex (Index=2)
ListBox SelectIndex (Index=2)
End Record on Friday, April 02, 2004 at 9:44:27
<Next Command>

AttachName: Index=2

	Value
ItemIndex	6

Use one of the following methods to record or view scripts in the Visual Navigator window:

- Choose **The Visual Navigator** option from TestPartner’s Welcome dialog box. TestPartner opens a new untitled script in the Visual Navigator window.

  The Welcome dialog box is an optional display that can be turned on or off using TestPartner’s General options (see “Customizing TestPartner’s Options” on page 26).

- With TestPartner already open, click the **Show Visual Script Window** button. TestPartner opens a new untitled script in the Visual Navigator window.

Configuring Record Options

TestPartner provides a standard set of Record options that control TestPartner's behavior while recording scripts. These options regulate the way TestPartner records specific actions performed on controls, the timings that are used, and hotkeys that are available while recording. Before recording a script, review Record options and, if necessary, make adjustments to suit user preferences. These selected options will determine how each user action is recorded.

To modify the Record options:

1. Click **Tools>Options**. The Options dialog box appears.
2. Double-click the **Record** option from the tree view. The Timing, Attach Name, Hotkeys, and Code Generation sub-categories appear in the outline view under the Record options. The Record options appear in the right pane. When the Timing, Attach Name, or Hotkeys categories are selected, the following subcategories appear under the Record option:

   ◇ **Timing —** Control the amount of time lapsed before pauses are recorded into the script.

   ◇ **Attach Name —** Control the items that are recorded using raw attach names, object map aliases, or user prompting.

   ◇ **Hotkey —** Determine the hotkey combinations that are used for inserting checks and events and for starting and stopping recording.

   ◇ **Code Generation Options —** Choose the format of the date to be added to the Begin and End Record Comments.

   A brief description of the option appears below the list of options when an item is highlighted. For more information about the various Record option settings, refer to TestPartner's online help.
3. Turn an option on or off by selecting the option and choosing **Yes** or **No** from the drop-down list. If the option requires a specific value, the option can be selected and its corresponding value edited.
4. Click **OK** to save settings and close the Options dialog box.

   These settings are only used in the active recording session. Optionally, click **Save As** in the **Record Settings** area to assign a group name to settings so they can be used in future record sessions.

Applying Record Settings

Once a group of record settings have been created and saved (see "Configuring Record Options" on page 45), they can be applied to any future recording session.

Use the following procedure to apply a saved group of record settings:

1. Click **Tools>Options**. The Options dialog box appears.
2. Double-click the **Record** option from the tree view.
3. In the **Record Settings** drop-down list, click the name of the group of settings to apply. Choose **System Defaults** from the list to restore TestPartner's original settings.

   TestPartner updates the current Record options with the settings in the saved group of options.
4. Click **OK** to use the selected settings in the record session and to close the Options window.

Configuring Record Options for Java Objects

Compuware's Active Object Recognition (AOR) Configuration utility provides custom support for how Java controls are recognized and recorded. It enables the use of supported classes to create definitions for Java controls without making hard code changes. This provides the flexibility to extend the testing tool's Java testing capability as the Java language evolves or as testing needs change.

Start the AOR Configuration utility by clicking the taskbar's **Start** button, then choosing **Programs>Compuware>TestPartner>Active Object Recognition>AOR Configuration Utility**. Refer to the AOR Configuration utility's online help for instructions. An AOR tutorial is also available to help familiarize you with using the utility to create custom definitions for Java controls. The tutorial uses the AOR Demo application, which contains a group of Java controls.

Recording New Scripts

Use TestPartner to record the actions performed on Windows objects or Web applications and to automatically generate a script. When a task is recorded and performed, the actions — and the responses of the application worked with — are translated into commands and pasted into a TestPartner script. While a script is recorded, everything done, except the interaction with TestPartner, will be recorded until the session is stopped. After recording is finished, the script can be modified. TestPartner can also be modified to record certain information about commands by creating Attach Name profiles. Refer to the TestPartner online help facility for more information about creating Attach Name profiles.

To begin recording a script:

1 If necessary, open a new script by right-clicking the **Script** button in the **Asset Types** pane and choosing **New Script**. The Asset Browser displays a new script in either the Visual Navigator window or the VBA Code window.

Note: Record in an existing script by opening the script and positioning the cursor at the script line where to begin recording (in the Visual Navigator window, this would be the node previous to where the new information is to be inserted).

2 Ensure that the target application or Web application is at the proper point to begin recording.

3 Click **Script>Record** to begin recording the script.

4 Perform the actions to be recorded into the script.

To suspend recording and insert checks or events into the script using shortcut keys:

- ◇ Press the **Insert Check** shortcut key, ALT+F8, to insert a check into the script and to automatically resume recording after the check has been created (see "Adding Checks While Recording Scripts" on page 53).
- ◇ Press the **Insert Event** shortcut key, ALT+F7, to insert an event into the script and to automatically resume recording after the check has been created (see "Adding Events While Recording Scripts" on page 68 for details).

5 Click **Script>Stop Recording** or click the **Stop Record** button on the toolbar.

TestPartner also stops recording when it is made the active window. When TestPartner is active, the script will contain the information necessary to playback the actions that were recorded.

Debugging Scripts

Because TestPartner uses VBA as its scripting language, all of the VBA debugging options are available. Code can be complied, stepped through, have watches added to it, and breakpoints can be set in it.

All of TestPartner's debugging options are available from the **Debug** menu on TestPartner's main menu.

For more information about debugging scripts, refer to the Visual Basic online help facility, which can be accessed by choosing **Microsoft Visual Basic Help** from TestPartner's **Help** menu.

Printing Scripts

Printing a script produces a hard-copy record of the target application's tests. The script can be printed exactly as it appears when viewed.

To print the script:

1 Click **Script** in the **Asset Types** pane. The Asset Browser displays a list of available scripts in the right pane.

2 Select the script in the right pane, and click **File>Print**. The Print dialog box appears.

3 Select a print range and click **OK**.

Playing Back Scripts

After a script is recorded it can be played back at any time. When TestPartner plays back the script, it drives the target application or Web application according to the actions recorded in the script. You can observe these actions taking place on the screen. A script can also be played back from the command prompt.

Because scripts are recorded on the object level — without reference to an object's position on the screen — script playback is also object-oriented. This means that TestPartner interacts directly with the Windows objects. It does not matter if there have been environmental changes or changes in the target application — scripts will still play back properly.

Whenever a script is played back, TestPartner repeats all of the actions that have been recorded in the script and conducts the checks that have been established.

To playback scripts.

1. With the script open, click **Playback Script** from the toolbar.

   A **Save Script** prompt appears. After saving the script, the Run Script dialog box appears. Enter a description that appears in the log after the script runs. The logging or playback options can also be changed before the script is run.

2. Click **OK** to play back the script. As the script runs, each recorded action — such as menu, radio button, and check box selections — are played back on the desktop or in the target application.

After the script completes playback, TestPartner becomes the active window. Depending on the selected options, you will either be returned to the open script or the script's log opens so that you can review the results of the test run.

Pausing Script Playback

Pausing a script temporarily halts the script playback. Pausing can be useful to make a correction in the target application or Web application before playback is continued. A script can be paused at any time during a playback session by clicking **Script>Pause**.

Note: This requires that TestPartner's desktop must be visible while playing back a script. If Minimize is selected in Playback options from TestPartner's General options, the script cannot be paused during playback.

Once a script is paused, a dialog box appears that enables you to end the playback or to continue playing back the script when ready.

Stopping Script Playback

Two ways to stop a script from playing back are:

- Press the **Stop Playback** shortcut key, ALT+F12. When the **Stop Playback** Hotkey is pressed, all script playback functions are stopped immediately.
- Click **Stop Script** on the toolbar.

Data-Driven Test

While testing a Windows-based or Web-based application, you may want to run a test repetitively, supplying different data each time the test is runs. For example, on a Web-based registration form, you may want to run a test several times to test the application's field-level verification capability. Using the Data Test Creation wizard, you can create a single test that enters an invalid zip code or phone number the first time it runs, an invalid password the next time it runs, and so on. This saves the time and effort of creating a new test for each required data set.

The Data Test wizard provides a non-programmatic way to create data-driven tests, which are usually used against form-based applications. The data files provide a convenient way of supplying input to TestPartner scripts. Using the Data Test wizard, you can record a test, choose the fields to include in the data file, and populate the data file with a data grid.

To access the Data Test wizard:

1. Open a script and choose **Create Data Test** from the **Script** menu. Follow the wizard's instructions.
2. To edit the data associated with a data test, choose **Modify Data Table** from the **Script** menu and select a data table from the Data Table List dialog box.

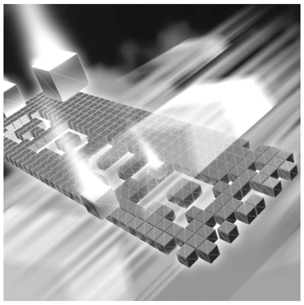
Running *QA*Run or WinRunner Scripts in TestPartner

TestPartner can launch *QA*Run or WinRunner scripts. Either *QA*Run or WinRunner must be installed on the same machine as TestPartner in order to execute their respective scripts.

After a tool domain is created for a *QA*Run or WinRunner script, the script can be inserted and executed from within a TestPartner script. Once a script has completed execution, log files are generated in both TestPartner and the script's respective product. When the TestPartner script is executed, the *QA*Run or WinRunner scripts will run.

*QA*Run and WinRunner scripts cannot be modified in TestPartner. *QA*Run and WinRunner scripts can only be modified within their original products. For information on modifying and running *QA*Run or WinRunner scripts, see the appropriate product's documentation.

Chapter 5
Checks



Using Checks

A check is a definition of the expected state of an aspect of the target application at a particular point. Typically, a check is carried out after the target application has performed a series of directed actions. The check determines whether the results of the actions are as expected. In short, checks determine whether the target application works.

You incorporate checks into scripts to verify that the target application is providing the anticipated responses to a set of inputs. Checks let you confirm that the application is validating input, performing calculations correctly, saving data reliably, and reporting accurately by comparing actual responses to expected responses and reporting any discrepancies. Automated testing is based on a simple premise — a correctly functioning system will respond in a known way to a known input.

Test designers specify the business scenarios to be tested. They identify functions that must be verified and any inputs that are necessary to accomplish this. It is done by defining actions that must be performed to test those functions and using TestPartner to convert those actions into script statements. Scripts enable you to repeat the tests on demand. However, to know whether the system actually works, you must define what you expect it to do in response to those actions and check that it actually does so.

An application's expected responses are stored within TestPartner as a *check*. A check is a definition of what the application should be displaying at any particular point. You can define a variety of check types. Each one contains special elements that enables you to test aspects of a target application or Web application. Table 5-1 briefly describes the check types that are available as you develop test scripts. Once you decide on the type of check to use, see "Creating Checks Using the Check Wizard" on page 55 or TestPartner's online help to assist you in completing the check's specific requirements.

Table 5-1. Check Types

Check Type	Function
Bitmap Check	Compares a bitmap in the target application with one that was previously defined. Bitmap checks verify the appearance of toolbars, the desktop, and other windows that contain non-textual information.
Clock Check	Measures the time the system takes to perform a process. Clock checks help check the application's performance under varying CPU or network loads.
Content Check	Verifies the content of lists and tables in a window or Web page. Use content checks to check the text of the items in a list, the number of items in a list, and whether a list item is selected. The content check can also be used to check the number of columns and rows in the table and to check the capitalization of any text in a table cell.
Field Check	Checks a defined area, or field, for specific information that matches the defined criteria. Field checks can be used to verify that information in a field matches a specific text, numeric, date, time, or pattern value.
Page Check	Page checks that were created with previous versions of TestPartner are now Property checks.
Property Check	Verifies the properties of the controls found on a dialog box or Web page. Use property checks to check the size and position of controls, to check their labels and IDs, and to determine if they are active, selected, or cleared. Property checks can also be used to check ActiveX control properties and to verify the HTML objects on a Web page.

Check Type	Function
Text Check	Validates the text in a window or Web page. Use text checks to verify the text in a window and to Entire windows can be checked and areas masked that contain variable data, such as dates, that are to be ignored during the check.

To verify, copy, rename, and delete checks, see TestPartner's online help.

Adding Checks to Scripts

You can add a check to a script while recording, or add a new check to an existing script. If you already know what type of check to add, want to insert an existing check, or want to create a check independent of a script, refer to the instructions in TestPartner's online help.

By default, TestPartner minimizes while you are recording. You can change this behavior using the General options. The following procedure assumes that TestPartner is minimized during recording.

Adding Checks While Recording Scripts

The easiest way to insert a new or existing check into a script while recording is to use the **Insert Check** shortcut key, ALT+F8. If the application you are testing already uses this key combination, reset the **Insert Check** hot key to another value (refer to the procedure in TestPartner's Online Help).

To add a check while recording a script:

1 Open a new script using TestPartner's Asset Browser.

2 Click the **Record Script** button to begin recording the new script. Continue recording until the point to insert the check is reached.

3 Press the **Insert check** shortcut key, ALT+F8. The Browse for Checks dialog box appears. It displays a list of existing checks. Use it to insert an existing check, to insert a check and open it for editing or viewing, or to create a new check.

4 Determine if the desire is to add a new check to the script or add an existing check to the script.

   ◇ To add an existing check to the script, select the check in the Browse for Checks dialog box and click the **Insert** button.

   ◇ To add a new check to the script, click the **New** button.

The Check Wizard starts and guides the process of check creation.

5 Click **OK** on the Check Wizard dialog box to begin using the wizard. TestPartner minimizes leaving an unobstructed view of the target application or Web application.

   A cross-hair cursor appears and the window objects are highlighted as the mouse moves over them in the target application or on the desktop.

6 Select the window object that contains the field to check. The **Check Wizard** dialog box appears. This dialog box displays the checks most likely to apply to the selected object. If a check is selected in the **Available Checks** list, a detailed description of that check type's purpose appears in the **Description** area.

7 Select the type of check to create and click **Create Check**. A new check is created for the specified type.

Note: For more information about creating specific check types, see "Creating Checks Using the Check Wizard" on page 55 or refer to TestPartner's online help.

8 Click **Save and Close** to save the check and insert it into the script. Once the check is saved, continue recording the script.

Adding Checks to Existing Scripts

Sometimes it is easier to record of the script automation in one session and to add any test assets to the existing script later. If this is the case, open a script for editing and then insert new or previously created checks into the script.

To add a check while editing a script:

1 Open the script in the VBA Code window using TestPartner's Asset Browser.

2 Navigate to the location of where to insert the check.

3 Click **Insert>Check>Check Type**. The Browse for <*Check Type*> dialog box appears. This dialog box only shows the checks of the type specified from the menu selection.

   ◇ To add an existing check to the script, select the check in the Browse for <*CheckType*> dialog box and click the **Insert** button.

   ◇ To add a new check to the script, click the **New** button.

Note: Add a new check to a script by clicking **Insert>Check Wizard** from the menu. This will start the Check Wizard, as described in "Adding Checks While Recording Scripts" on page 53.

Creating specific check types is discussed in more detail in "Creating Checks Using the Check Wizard" on page 55 or in TestPartner's online help.

4 Select the appropriate button to insert the check into the script.

Creating Checks Using the Check Wizard

Checks can be defined while recording a script or set up "off-line" by defining them directly and adding them to the Asset Browser. There are several types of checks that can be defined and inserted into a script. Each check type is described in more detail in the sections that follow.

Defining Bitmap Checks

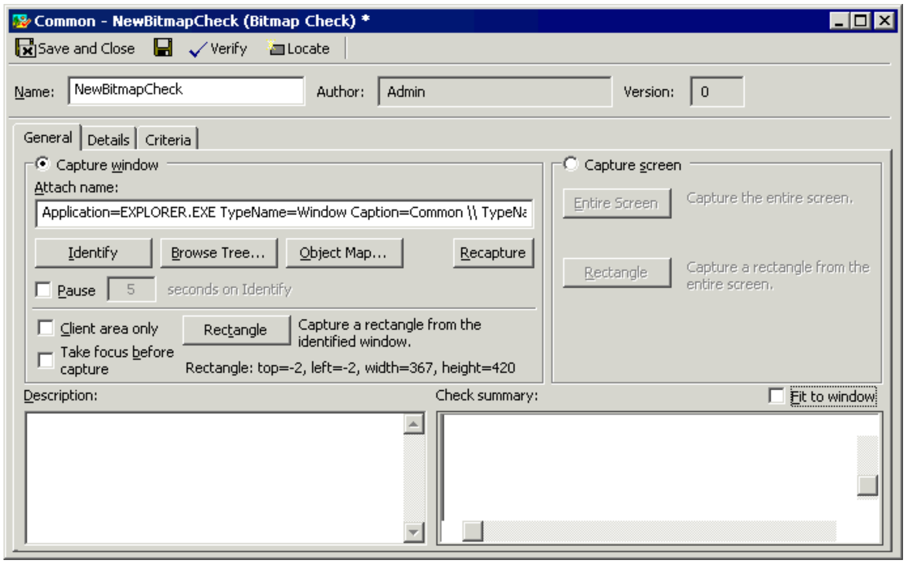
Bitmap checks enable you to verify the appearance of a bitmap image. These checks are used to check the appearance of toolbars, the desktop, and other windows that contain non-textual information. When creating the check, capture the image within a rectangular area of the screen.

The coordinates of the area that you capture can be defined relative to a particular window or as absolute screen coordinates. You can also define one or more areas to ignore, which enables you to ignore image areas that contain legitimately variable data (such as a clock display) when the verification is done.

The Check Wizard contains the following tabs:

General Tab: An important tab when creating bitmap checks. The General tab allows you to define how the bitmap image will be captured (see Figure 5-1).

Figure 5-1. The Check Wizard's General tab for Bitmaps



Name: NewBitmapCheck Author: Admin Version: 0

(•) Capture window

Attach name:

Application=EXPLORER.EXE TypeName=Window Caption=Common \\ TypeNa

[ ] Pause 5 seconds on Identify

[ ] Client area only

[ ] Take focus before capture

Capture a rectangle from the identified window.

Rectangle: top=-2, left=-2, width=367, height=420

( ) Capture screen

Capture the entire screen.

Capture a rectangle from the entire screen.

Description:

Check summary:

Control	Description
Capture window radio button	Click to capture images based on their entire object size
Capture screen radio button	Click to capture images based on absolute screen coordinates
Client area only checkbox	Click to exclude menus and scrollbars in the captured object image
Take focus before capture checkbox	When selected, focus is removed from the object before capture in order to exclude highlighting and cursor position
Rectangle button	Click to define an image area within the selected object

Details Tab: Use the Details tab to exclude areas from the bitmap check. These areas are ignored when the check is played back. To exclude an area, click **Add**. Then in the **Captured bitmap** window, drag the cursor over the area to exclude. The excluded area is masked. Add as many areas as necessary.

Criteria Tab: Use the Criteria tab to control how the bitmap is evaluated. You can specify the number of mismatched pixels that will cause the bitmap check to fail.

Defining Clock Checks

Clock checks enable you to verify how long a process takes to complete. They are used to carry out performance checks on the target application or Web application. Use clock checks to determine whether, under controlled conditions, the target application performs tasks within predefined response times. When creating the check, specify an acceptable response time. When the check runs, the system's actual response time is recorded and compared to the specified time. If it is within the acceptable range, the check passes; if it is not within the acceptable range, the check fails.

Clock checks are created and inserted into the script differently than TestPartner's other checks. Clock checks are not only defined using the Check Wizard, but also they require that you insert specific code into the script in order for the clock check to use the "stopwatch" functionality.

Note: You must be working in the VBA Code window when you are inserting clock checks.

To create both the "clock object" and clock check in the script:

1 Set the cursor before the process to check.

2 Create a clock object by inserting the following code:

```
Dim t As TClock
Set t = Clock("Check name")
t.Reset
t.Start
```

   Where `"Check name"` is the name of the clock check to create.

3 Locate the end of the process to check. Stop the clock by inserting the following code:

```
t.Stop
```

4 Click **Insert>Check>Clock**. The Browse Clock Checks dialog box appears.

5 Click **New**. A new clock check setup appears in the TestPartner window.

6 Type a name for the clock check in the **Name** field. The name must match the name used when the clock object was created in step 2.

7 On the **Details** tab, select the operation, such as < **less than**, and type the value to apply to the clock check.

8 Click **Save and Close** to save the check and insert it into the script.

Defining Content Checks

A content check enables verification of the contents of table and list controls in a Windows-based or Web-based dialog box or a Web page. List controls include list boxes, drop-down lists, combo boxes, and tree controls. The desktops on Windows 95 and Windows NT 4 are also list controls.

When you create a content check for a table or list control, TestPartner captures the content of that table or list control. For example, if you are capturing a list box, TestPartner captures all the items in the list box. The content check displays a list containing each item and indicates whether the item was selected in the list when the check was captured. You can edit the list items, add or remove items, change the selection state, and change the order that items appear in the list. When the check is verified, the list is captured and compared to the defined version. If the two lists match according to the criteria defined, the check passes. If they do not match, the check fails.

When using a content check to check tables, you can also check the number of rows and columns in the table and the capitalization of the text in each table cell.

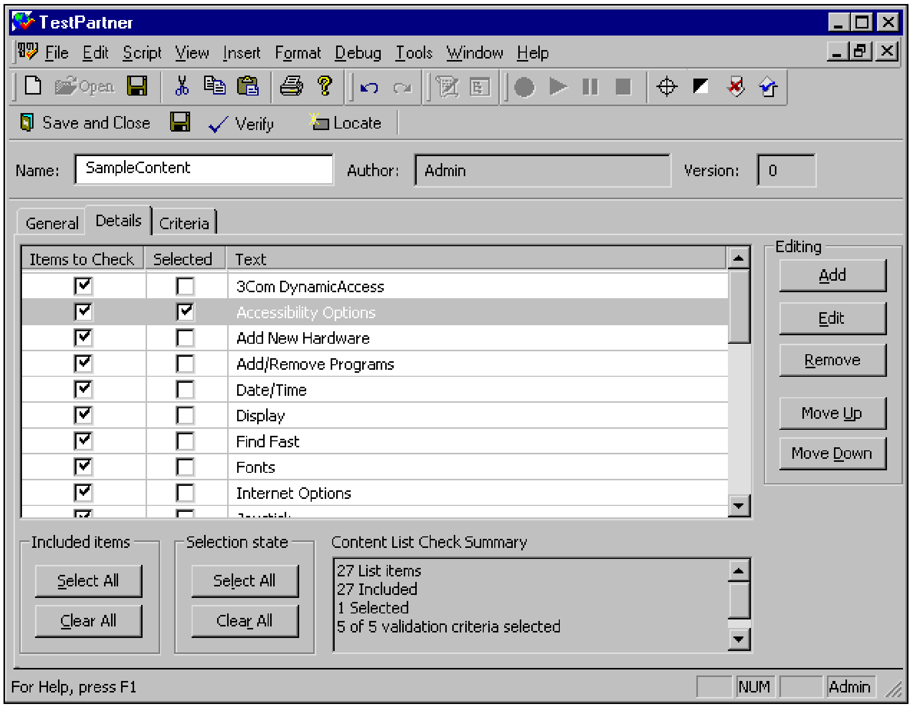
When using the content check to check list controls, you can check the capitalization of the text, the number of items, the positions of the items, which items are selected, and the text of all list items.

The Content Check Wizard contains the following tabs:

General Tab: Similar to other checks' General tabs, except this General tab contains capture settings.

Details Tab: Displays all of the table or list items and enables you to edit and arrange the values (see Figure 5-2).

Figure 5-2. The Content Check Details Tab for a List Box



Name: SampleContent Author: Admin Version: 0

Items to Check	Selected	Text
☑	☐	3Com DynamicAccess
☑	☑	Accessibility Options
☑	☐	Add New Hardware
☑	☐	Add/Remove Programs
☑	☐	Date/Time
☑	☐	Display
☑	☐	Find Fast
☑	☐	Fonts
☑	☐	Internet Options

Content List Check Summary

27 List items
27 Included
1 Selected
5 of 5 validation criteria selected

For Help, press F1

Control	Description
Add button	Click to add items to the list
Edit button	Click to edit existing items
Remove button	Click to remove items from the list
Move Up button	Click to move items upward in the list
Move Down button	Click to move items downward in the list

All of the items captured during the creation of the check can be edited or arranged from this tab. The summary area provides an overview of the items included in the check.

The information that appears on the Details tab is different from content checks on tables. Refer to TestPartner's online help for specific information about using the content check to check tables.

Criteria Tab: Contains the settings that are used to evaluate all items in the content check. You can specify whether the list or table item's capitalization, total number of items, position of items, selection state, or exact item text are considered when the check is verified.

Defining Field Checks

Field checks enable you to conduct specific comparisons on the text of a dialog box or Web page. You can compare dates and times, item text, numbers, and pattern values for individual fields selected in a window or area.

Field checks provide advantages over the standard text check. Like text checks, field checks enable you to verify that required text is present in the target application. They also enable you to verify specific types of data, such as numbers or dates. For example, use a field check to verify that a value falls between a lower and upper limit or if a particular area of the screen contains today's date. You can create field checks that verify the following data:

- ASCII values
- Numeric values
- Date values (fixed and aged)
- Time values
- Patterns

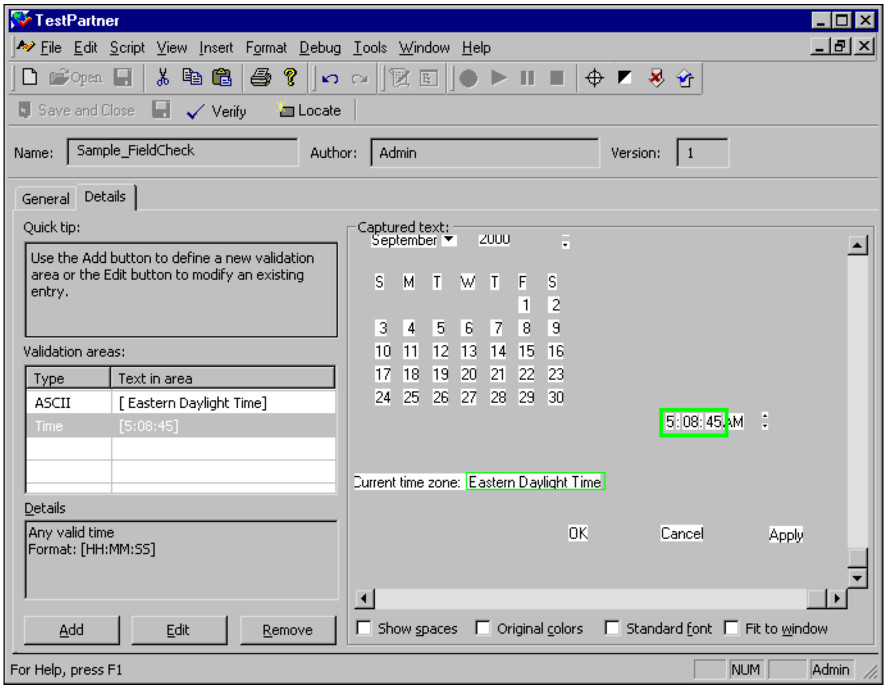
When creating a field check, you capture the text from an existing window or Web page. You can edit the captured details and add or remove items before saving it. When the check is verified, the text in the actual dialog box or Web page is captured and compared to the defined version. If the significant details of the two match exactly, the check passes. If they are different, the check fails.

The Field Check Wizard contains the following tabs:

General Tab: Similar to the General tab for other checks, except this General tab provides the **Auto Concatenate** capture setting that allows to you treat text items that are adjacent to one another as one block item.

Details Tab: Displays the text in the captured window and enables you to define specific fields to validate (see Figure 5-3).

Figure 5-3. The Field Check Details Tab



Name: Sample_FieldCheck Author: Admin Version: 1

Quick tip:
Use the Add button to define a new validation area or the Edit button to modify an existing entry.

Validation areas:

Type	Text in area
ASCII	[Eastern Daylight Time]
Time	[5:08:45]

Details
Any valid time
Format: [HH:MM:SS]

Captured text:
September 2000

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

5:08:45 AM

Current time zone: Eastern Daylight Time

Control	Description
Add button	Click to add a validation area to the check
Edit button	Click to edit the selection in the **Validation areas** list
Remove button	Click to remove validation areas from the check

To add a new validation area, position the cursor over the field and click. The Area Definition dialog box appears, in which to select the type of area being created: ASCII, Numeric, Date, Aged Date, Time, or Pattern. For more information about defining the criteria, refer to TestPartner's online help.

Defining Property Checks

Property checks verify the properties of the controls on a dialog box or Web page. You can check the size and position of each control, their legends and IDs, and whether they are active, disabled, selected, or cleared. You can check a single control, or check several controls within an application window.

The Property Check Wizard contains the following tabs:

General Tab: Similar to the General tab for other checks, except this General tab provides the **Include menus** and **Include hidden objects** capture settings. If these items are selected, menus and any non-visible objects are captured and considered for the check.

Details Tab: Displays all the properties on the control that you captured and enables you to edit the control's values. Click a property name to select it for processing, or hold down the CTRL key to select multiple properties for processing. You can also double-click a property's value to edit it.

To ignore certain properties when the check is verified, select the properties, then right-click and select **Exclude Selected Properties** from the shortcut menu. Excluded properties display with gray backslashes.

Defining Text Checks

Text checks enable you to verify the textual content of the target application's displays. Similar to the field check, the text check also captures the text that appears in the window. The text check, however, enables you to specify areas that are ignored when the check is run. You might want to ignore areas on the display that contain information that is expected to change, such as dates or times, that you do not want to consider in the check.

Unlike bitmap checks, which compare the appearance of an area of the screen with an expected appearance, text checks read the displayed data as strings. You can use the resulting strings to perform more sophisticated checking.

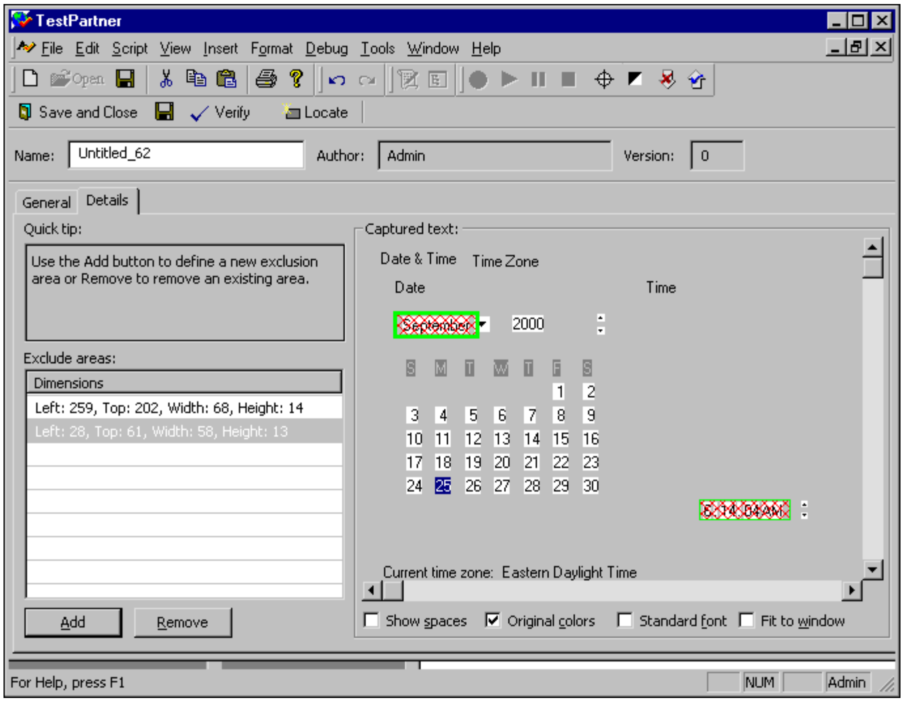
When you create a text check, you capture text from a window on the screen. The window name and the areas to ignore are recorded. When the check is verified, text from the screen is captured and compared to the defined version. If the significant details of the two match exactly, the check passes. If they are different, the check fails.

The Text Check Wizard contains the following tabs:

General Tab: Provides the **Auto Concatenate** capture setting that treats text items that are adjacent to one another as one block item.

Details Tab: Displays the text in the text window and enables the defining of specific fields to validate (see Figure 5-4).

Figure 5-4. The Text Check Details Tab



Name: Untitled_62 Author: Admin Version: 0

Quick tip:

Use the Add button to define a new exclusion area or Remove to remove an existing area.

Exclude areas:

Dimensions
Left: 259, Top: 202, Width: 68, Height: 14
Left: 28, Top: 61, Width: 58, Height: 13

☐ Show spaces ☑ Original colors ☐ Standard font ☐ Fit to window

For Help, press F1

To add a new Exclude area, drag the cursor over the area to ignore and click. When the check is verified, the item is masked so that it will be excluded (ignored).

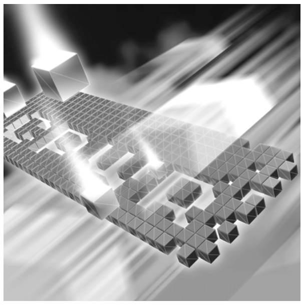
Verifying Checks

Checks can be verified manually or during script playback. If the check fails — that is, if the actual state of the target application does not match that specified in the expected check definition — TestPartner records the actual state so you can view the differences.

When a check is encountered during playback of a script, the actual state of the target application is compared to the expected check definition. If logging is enabled, and there are no material differences, a “Pass” is written to the log. If there are material differences, a “Fail” is written to the log and the actual state of the target application is recorded in the log for reference.

The reasons for the check failure can be viewed from the log. By default, checks that pass are displayed in green, and checks that fail are displayed in red. Double-click the Check Name column of the failed check from the Browsing Logs dialog box to view the differences.

Chapter 6
Events



Using Events

TestPartner uses events to monitor the state of the target application or Web application and your entire computer system. Events are often fundamental components to the success of your test system. They can be used to synchronize your scripts with the responses of your target application. You can use events to make test scripts recognize and appropriately respond to error conditions that may occur within the system during testing. If used properly, events help you create intelligent scripts, which are capable of controlling the flow of automation based on external conditions.

Operating a computer application often entails performing particular tasks at certain times. When you carry out those tasks, you perform a few actions and then wait for the system to process your input before continuing with the next operation. In order to be effective, your test scripts must be capable of behaving in the same way. For example, two conditions that your script must be able to accommodate are varying system response times and error conditions.

All networked computer systems experience variable response times caused by varying loads on the server and network. Even single-user systems have varying responses. Your scripts must allow time for the target application to process its input before continuing with the next input. You could achieve this by pausing after every line in your script — to allow the target application sufficient time to process the last command. But, pausing for a fixed time is inefficient and unreliable.

If the pause is based on the best possible response time on a heavily loaded system, the script would continue before the target application was ready. In contrast, if the pause is based on the worst possible responses for a lightly loaded system, unnecessary time is wasted when the script waits for the target application.

Similarly, if an error occurs, your script must be able identify and respond automatically. Sometimes, error conditions occur for reasons unrelated to the testing process itself — the connection to the server may be broken, a network system message may interrupt the process, or an e-mail message may be received. Your script must be able to recognize and respond to error conditions before continuing the test.

The most efficient and reliable way to drive a computer application is to watch the screen for an indication that the target is ready to continue (or that an error has occurred). TestPartner can accomplish this by monitoring the target application and confirming that it is ready before continuing. It can recognize unexpected conditions and deal with them.

The conditions that TestPartner must respond to are called events, and the definition of an event is stored in TestPartner as an asset in the Asset Browser. An event is a condition (external to TestPartner but internal to the computer system) that can be tested by the script. An event can belong to any of the categories described in Table 6-2:

Table 6-2. Event Types

Event Types	Function
Bitmap Event	Monitors for a graphic to be present or absent.
Date/Time Event	Monitors for a certain date and/or time (according to the PC's internal clock) to be reached or to elapse.
Key Event	Monitors for a particular keystroke combination to be typed.
Menu Event	Monitors for a specific menu item to be selected.
Mouse Event	Monitors for a mouse button to be clicked or released within a certain window.
Screen Event	Monitors for a sequence of characters to appear somewhere within a window.
Window Event	Monitors for a window to be created, destroyed, moved, sized, or otherwise changed.

Knowledge of events enables a script to become intelligent, and to make decisions about the flow of the automation process depending on the state of external conditions. When an event has been defined, your script can:

- ◇ Test if the event has taken place
- ◇ Wait until it occurs before continuing
- ◇ Perform a task whenever it occurs

When events are used within a TestPartner script, they are set as conditions that either wait until a specified event becomes true before continuing or, alternatively, perform a series of predefined instructions whenever a specified event becomes true.

Choosing Wait or Whenever Statements

Once you determine the type of event to use, you must determine how you want the event to act. Events can wait for or react to conditions whenever they occur. When you insert an event into your script, TestPartner asks you whether to insert it as a Wait statement or as part of the Whenever statement. TestPartner formats the event based on your response.

Waiting for Events

TestPartner waits for a specified event to occur before proceeding. For example, if you were explaining how to use a computer system, you might use a series of instructions that explain the interaction between the user and the computer. For example:

```
When that message appears, type this ...
Wait until you see that icon, and then click there ...
After the confirmation dialog box appears, select the File menu
...
```

This type of instruction is considered event-driven; you perform an action, wait for an event to occur, and then perform the next action. Script events work the same way. The script simulates a user action and then waits for confirmation from the application before it continues.

Waiting for a system login prompt is an example of an event that is used used with a Wait statement. In this example, when your script is running against a network-based application that requires the user to log in, the amount of time it takes for the login prompt to appear often varies. To account for this, you can insert the event as a Wait statement that instructs your script to wait for the login prompt before proceeding to type a name and password.

TestPartner's General options control the amount of time for which script waits for an event and the directions on how to proceed.

Acting Whenever Events Occur

The Whenever statement tells TestPartner to watch for a specific set of occurrences and, if one of them occurs, perform a special set of steps. The Whenever statement is useful for trapping unexpected error conditions during a test run. For example, you can include events in your scripts that recognize when the connection to the server has been interrupted by a communications error or when a network system message is received. If the event uses the Whenever statement, the script can report the problem or proceed with a work-around.

The Whenever statement enables a script to perform a series of instructions whenever a particular event occurs. When a Whenever statement is encountered in a script, the event is placed in TestPartner's active event list. Script execution continues, but TestPartner continuously monitors the system to determine if the event has occurred. If the event is detected, the normal program flow is interrupted and control passes to the Whenever function. Upon completion of that function, normal program flow is restored, and the script continues from the point at which it was interrupted. The event stays in the active event list and can trigger the Whenever function again.

Adding Events to Scripts

Events are test assets that are stored independently of the script. If you have an event that you already created, you can add it to the script while you are recording or while you are editing the script. Likewise, you can use the same procedure to simultaneously add an event and create its definition while you are recording or editing the script.

Adding Events While Recording Scripts

The easiest way to insert a new or existing event into a script while recording is to use the Insert Event shortcut key, ALT+F7. If the application you are testing already uses this key combination, you will need to reset the Insert Event shortcut key to another value (refer to TestPartner's online help).

Use the following procedure to add an event while recording:

1 Open a new script using TestPartner's Asset Browser.

2 Click the **Record Script** button to begin recording the new script. Continue recording until insertion point of the event.

3 Press the **Insert Event** shortcut key, which is assigned to ALT-F7 by default. The Browse for Events dialog box appears. This dialog box displays a list of existing events. Use it to insert an existing event, to insert an event and open it for editing or viewing, or to create a new event.

4 Determine if the desire is to add a new event to the script or add an existing event to the script.

   ◇ To add an existing event to the script, select the event in the Browse for Events dialog box and click the **Insert** button.

   ◇ To add a new event to the script, click the **New** button.

   When a new event is created, select the type of event from the Insert Events dialog box's drop-down list and click **OK**. The Event Wizard starts and guides the process of event creation.

   The creation of specific event types is described in more detail in the "Creating Events Using the Event Wizards" on page 70.

5 After the event is created or selected, determine if the desire is to insert the event using Wait or Whenever code (see "Choosing Wait or Whenever Statements" on page 67 for more information about the differences).

   Select the appropriate button to insert the event into the script. Because the event was inserted while recording, TestPartner returns to the active record session to continue recording the script.

Adding Events While Editing Existing Scripts

Sometimes, you might find it easier to record all your script automation in one session and add any test assets to the existing script later. If this is the case, you can open a script for editing and then insert new or previously created events into the script.

Use the following procedure to add an event while editing a script:

1 Open the script in the VBA Code window using TestPartner's Asset Browser.

2 Navigate to the insertion point of the event.

Note: If the script is open in the Visual Navigator window, select the node before which the event will be inserted.

3. From the **Insert** menu, choose **Event>Event Type**. The Browse for <*Event Type*> dialog box appears, and displays the events of the type specified.

   ◇ To add an existing event to the script, select the event in the Browse for <*EventType*> dialog box and click the **Insert** button.

   ◇ To add a new event to the script, click the **New** button.

     When a new event is created, select the type of event from the Insert Events dialog box's drop-down list and click **OK**. The Event Wizard starts and guides the process of event creation.

     The creation of specific event types is discussed in more detail in the "Creating Events Using the Event Wizards" on page 70.

4. After the event is created or selected, determine if the desire is to insert the event using Wait or Whenever code (see "Creating Events Using the Event Wizards" on page 70 for more information about the differences).

   Select the appropriate button to insert the event into the script, and continue editing the script.

Creating Events Using the Event Wizards

Events can be defined while recording a script or they can be set up "off-line" by defining them directly and adding them to the Asset Browser. There are several types of events that can be defined and inserted into the script. However, most events are created and defined with a similar procedure. Use the following procedure to define most event types and then refer to the subsequent sections to assist in completing the specific event-type information.

To create an event:

1. Insert an event into a script either while recording or while editing (see "Adding Events to Scripts" on page 68 for instructions).

2. Click the **New** button from the Browse For <*Event Type*> dialog box. The Define Event Name dialog box appears.

3. Type a name and description for the event and click **Next**. The Identify dialog box appears.

4. To confine the event action to a window or module, click the **Identify** button. TestPartner minimizes to leave an unobstructed view of the target application.

Position the mouse pointer over the required window and click. When TestPartner is restored, the window or module names appear in the dialog box.

As an alternative to using Identify, click the **Browse Tree** button to select an attach name from a list of current objects, or click the **Object Map** button to select an attach name from the existing Object Map entries.

5. Click **Next**. A dialog box that is specific to the type of event being created appears. See the appropriate event descriptions that follow for information about completing this dialog box.
6. Click **Finish** to complete the event's definition and choose determine to insert the event using Wait or Whenever code (see "Creating Events Using the Event Wizards" on page 70 for more information on the differences).

Defining Bitmap Events

A bitmap event detects the presence, absence, or state of a graphic in a window. For example, you could use a bitmap event to synchronize the test script with the Web application by waiting for the absence of a "busy" indicator.

The Bitmap Event dialog box enables you to capture the bitmap and specify whether TestPartner waits for the bitmap to appear or disappear from the specified area of the target application's window. See TestPartner's online help for more information about defining bitmap events.

Defining Date/Time Events

Date/Time events enable you to define date or time condition. TestPartner recognizes the event by monitoring the PC's internal clock.

The Define Date and Time dialog box enables you to define your date/time criteria. The dialog box is split into two panes:

- The **Time** pane is designed for scheduling an event at a given time (this event is typically used in a scenario where a script is executed often).
- The **Date** pane is designed for scheduling an event on a given date (this event is typically used in a scenario where a script is executed only once). See TestPartner's online help for more information about defining date/time events.

Defining Key Events

Key events allow your script to monitor actual keystrokes. You can use key events to:

- ◇ Build shortcut keys to make TestPartner perform an action whenever the shortcut key is used.
- ◇ Interrupt a script to take manual control of the target application.
- ◇ Pause a script until a particular key is pressed.
- ◇ Prevent certain keys from being entered.

Key events refer to actual keystrokes made by a user. Keystrokes that TestPartner types into an application during playback are not detected. When you define a key event, you can use the Identify dialog box to set the scope of the key event. For example, you can confine detection of the event to the identified window, to all windows belonging to the identified application, or apply it to any window.

You can detect any single key or "modified" key (a key used with any combination of the **{SHIFT}**, **{CTRL}** and **{ALT}** keys). You may also detect the use of any key pressed in the target application by listing **{AnyKey}** as a keystroke. See TestPartner's online help for more information about defining key events.

Defining Menu Events

Menu events are actual menu selections made by a user while the script is playing back. Menus that are selected by TestPartner during script playback are not detected.

You can choose to detect the menu item by its text or by its numeric ID. If you select this option, consider the application you are testing. Menu text is easier to read — but, because the event is based on the menu's text, you couldn't use the same event definition in multi-lingual versions of the application. If the target application has multi-lingual versions, you should probably detect the menu by ID. ID detection works well unless the application that you are testing dynamically assigns menu IDs. If this is the case, you can not detect the menu by its ID, because the ID will be different each time the menu is selected. See TestPartner's online help for more information on defining menu events.

Defining Mouse Events

Mouse events can be defined to react to mouse actions made by an actual user during script playback. Mouse actions that are made by TestPartner during script playback are not detected. See TestPartner's online help for more information on defining mouse events.

Defining Screen Events

A screen event detects the presence, or absence, of text in a window. Screen events are commonly used to synchronize with host-based applications and to detect error messages.

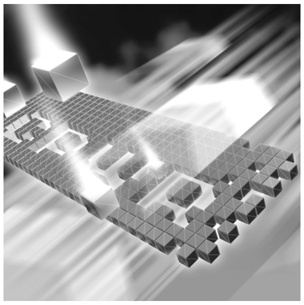
The Screen Event dialog box enables you to define the text you want to detect. If the text is currently displayed on the target application's screen, click the **Capture** button. If the text is not displayed, you can type the text in the edit box. See TestPartner's online help for more information on defining screen events.

Defining Window Events

A window event monitors a specific window's state. For example, a window event can be used to wait for a window to be created, destroyed, or moved.

Once the name of the window is defined using the event's Identify dialog box, the window state that the event is waiting for can be defined using the Window Event dialog box. See TestPartner's online help for more information on defining window events.

Chapter 7
Logs



Using Logs

TestPartner generates logs to record the results of an automated test run. A log records all the actions performed by the script and the results of the checks and events performed on the target application. Logs can be used to provide proof of the testing process and a permanent record of the target application's functional state at any given time.

One of the primary advantages of automated testing is that it doesn't require your presence while the test is running. But, even if you're not there, you still need to know the outcome of the test. You can accomplish this by generating logs for each test run. TestPartner can provide a detailed log for each test, so you can quickly and easily analyze the outcome of the tests.

While the script tests the target application or Web application, TestPartner writes all actions and results to a log file. Each script line that involves the target application generates a log entry. If a check fails (that is, the target application did not respond to TestPartner's input as expected), both the expected and actual responses are stored in the log for comparison. You can browse the log to display the detailed results of a TestPartner test script.

Logs contain plenty of information including the script name, the date and time each line of the script was executed, the command that was issued, the result of checks, and the machine that the script was executed from. Because so much information is available for each log, TestPartner allows you to customize the information. You can view only the categories that you are interested in — in order of preference. Once you've established your preferences, you can save your choices as a layout and apply it to other logs.

You can also apply filtering to your view of the log. Filters allow you to select the log entries that are of most interest. For example, choose to show only the results of checks or events to gain a quick overview of how the test ran.

TestPartner provides a toolbar exclusive to working with logs. Table 7-1 provides an overview of each button

Table 7-1. Buttons in the Logs Toolbar

Log Toolbar Button	Usage
	Customize Header: Adds, removes, and rearranges columns in the log display. Find procedures for customizing columns in TestPartner's online help.
	Group By: Arranges log information based on groups of information (i.e., script name, results, machine number). Find procedures for displaying log items by group in TestPartner's online help.
	Sort By: Arranges log information based on sort orders. Allows further qualification of the Group by order. Find procedures for sorting log columns in TestPartner's online help.
	Export: Exports the log from TestPartner into a text file. This file can then be imported into a source control program. Find procedures for exporting logs in TestPartner's online help.
	Preview: Displays the view of the log as it will print.
	Print: Prints the log without displaying a preview. This option also enables printer specification and setup information.

Log Toolbar Button	Usage
Save...	**Save Failed Checks as Expected**: To prevent checks in the log from failing, TestPartner can create a new version of each check in the log's script with the actual result of the check saved as the expected result. Find procedures for saving failed checks as expected in TestPartner's online help.
Filter...	**Filter By:** Enables selection of a filtered view from the drop-down list or to create, to remove, or to modify an existing filter.
Layout Manager...	**Layout Manager:** Enables saving and assigning a name to a specific column layout.

Creating Logs

TestPartner logs provide a valuable way to review the test results and analyze areas where the test may have failed. You can also use logs to record and document your testing procedures; however, it may not be necessary to log all script activity while you are still developing and testing your scripts. TestPartner also provides a way for you to turn logging off while developing and testing your test scripts.

To enable or disable logging activity:

1. Click **Tools>Options**. The Options dialog box appears.
2. Double-click the **Playback** option from the tree view and click **Logging**. The right pane of the dialog box displays the available logging options.
3. Select **True** from the **Logging** list to enable script logging.

   Control logging activity by setting options to determine if all activity or only error information is logged, if logs are appended to one another, and if the log number is automatically incremented each time the log runs. For more information about these options, refer to TestPartner's online help.

   Assign logs a log names. The log name is assigned to a collection of logs. For example, if a script runs three times during the development process, the script run can be associated with a log name so that the results are stored in the log with a logical association.

4 Type a name in the **Log name** field. This log name appears in the **Name** column when the **Log** button is selected from the Asset Browser.

5 Click **OK** to save settings and close the Options dialog box.

When running the script, a log will be generated using the assigned log name. If you did not enter a log name, the script name will be used instead. You can set TestPartner to automatically display the current log after a script is run, or you can view any log from the Asset Browser (see "Viewing Logs" on page 78 for information about viewing logs).

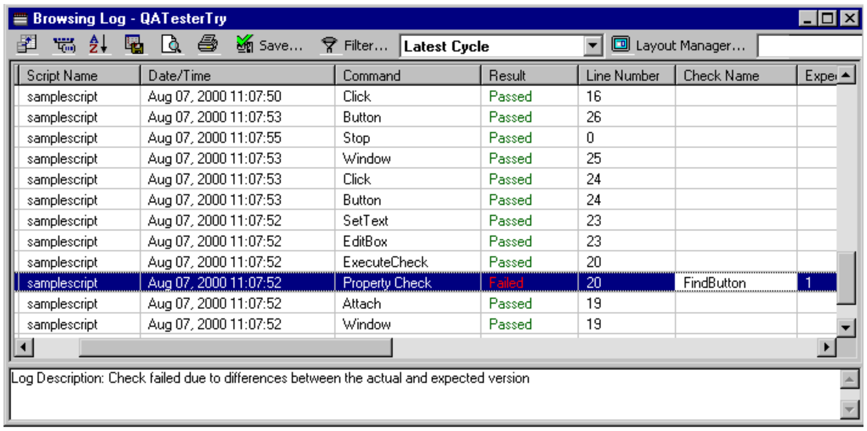
Viewing Logs

Similar to other TestPartner components, logs are stored in the Asset Browser.

To view a log from the Asset Browser:

1 From the Asset Browser, click the **Log** button in the **Asset Types** pane. The Asset Browser displays a list of available logs in the right pane.

2 Double-click a log name to view its contents. The log opens and displays the results of the test run as shown in the following example.



Script Name	Date/Time	Command	Result	Line Number	Check Name	Expe
samplescript	Aug 07, 2000 11:07:50	Click	Passed	16		
samplescript	Aug 07, 2000 11:07:53	Button	Passed	26		
samplescript	Aug 07, 2000 11:07:55	Stop	Passed	0		
samplescript	Aug 07, 2000 11:07:53	Window	Passed	25		
samplescript	Aug 07, 2000 11:07:53	Click	Passed	24		
samplescript	Aug 07, 2000 11:07:53	Button	Passed	24		
samplescript	Aug 07, 2000 11:07:52	SetText	Passed	23		
samplescript	Aug 07, 2000 11:07:52	EditBox	Passed	23		
samplescript	Aug 07, 2000 11:07:52	ExecuteCheck	Passed	20		
samplescript	Aug 07, 2000 11:07:52	Property Check	Failed	20	FindButton	1
samplescript	Aug 07, 2000 11:07:52	Attach	Passed	19		
samplescript	Aug 07, 2000 11:07:52	Window	Passed	19		

Log Description: Check failed due to differences between the actual and expected version

Checks that pass display in green and checks that fail display in red.

The log can also be used to navigate to other useful information about a script:

◇ To view the script that generated a log entry, double-click any line of the log, except lines that detail check results.

- ◇ If a check fails during the test, its expected and actual results are recorded in the log so differences can be compared. To view a failed check, double-click the entry in the log.

Submitting Defects to TrackRecord

TrackRecord is Compuware's defect-tracking tool for automated testing. You can use TestPartner in conjunction with TrackRecord to track bugs in your test application. For example, if a failed TestPartner test reveals a defect in the application, you can submit the defect right from TestPartner. TestPartner automatically enters relevant test information in the TrackRecord defect item, which reduces the amount of manual data entry. The failed test is linked to the TrackRecord defect by means of a defect ID.

To submit a defect:

1. From the Asset Browser, click the **Log** button in the **Asset Types** pane. The Asset Browser displays a list of available logs in the right pane.
2. Highlight the log that contains the defect and click **Tools>TrackRecord>Submit Defect**.

   If TrackRecord is not already running, TestPartner starts it. There may be a prompt to enter a TrackRecord logon password.

   In TrackRecord, a new Defect window is automatically opened. The TestPartner log data is automatically added to the Description field, where it can be edited.
3. Enter the appropriate data into the remaining fields of the defect. See TrackRecord's online help for further assistance.
4. Click **Save and Close** and return to TestPartner.

   The defect is added to the TrackRecord database. For information about submitting defects during testing or changing the TrackRecord login, refer to TestPartner's online help.

Index