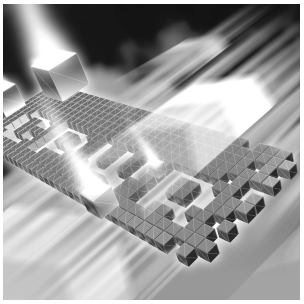
AOR Tutorial



This tutorial demonstrates some of the basic functions of the AOR utility. It takes you through the process of configuring and using the AOR utility. The code samples and class files necessary to run this tutorial are provided for you.

All class files for this tutorial can be found in Program Files\Compuware\Active Object Recognition\AORHelpers.

All Java source files for this tutorial can be found in Program Files\Compuware\Active Object Recognition\AORHelpers\Source.

In this tutorial, use TestPartner to run a test against a demo application. The demo application is a basic calculator written in Java. TestPartner cannot initially identify the calculator's controls properly. Use the AOR Configuration utility to add various levels of support for the calculator's controls. After you add each type of support, test the application using TestPartner. You should notice improved support for the demo application.

Prerequisites For Using the AOR Utility

The AOR Configuration utility configures Java objects for testing. It requires a knowledge of Java development. If you are using the AOR utility, the following prerequisites apply:

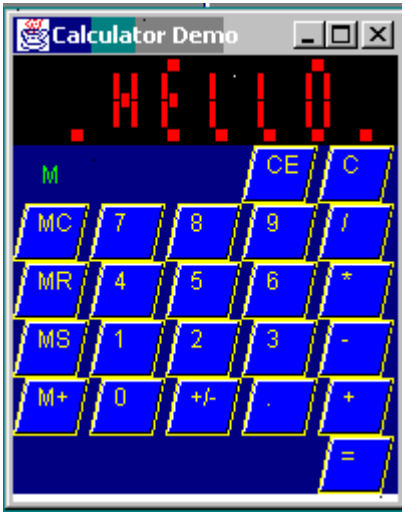
- The AOR Configuration utility is designed for use by Java developers and QA Analysts with Java development experience.
- The AOR Configuration utility may require some Java programming.
- If you do not have Java development experience, work with your development team to implement custom Java testing support.

- You must be familiar with your testing tool. TestPartner's online tutorial provides practical experience with using these two testing tools.

Starting the Demo Application

To start the demo application, follow these steps:

- Click **Start>Programs>Compuware>TestPartner >Active Object Recognition>AOR Demo.**



The demo application is a basic calculator, written in Java. Leave this demo application running for the duration of the tutorial unless you receive a message requiring you to restart the application.

Setting Up the Tutorial

For this tutorial, you will use the testing tool to record against this sample application, and identify some its controls.

1. Start TestPartner.
2. Click the **Identify** button on the TestPartner toolbar.

3. On the **Identify** dialog box, click **Identify**. A pointer appears.

4 Use the pointer to point to a button in the demo application. In the Identify dialog box, you will see that the type is not listed as a button. The testing tool does not properly support most of the controls in this application. Depending on how the testing tool is configured on your machine, the controls are interpreted as mouse clicks, bitmap selects, or text selects.

For more information on using the Identify tool and running scripts, see the TestPartner user documentation.

Building Basic Support

In this section, use the AOR utility to configure some basic settings and add support for the controls in the demo application.

- Click **Start>Programs>Compuware>TestPartner >Active Object Recognition>AOR Configuration Utility** to start the AOR Configuration utility.

  The AOR utility main screen appears.

Creating a Bundle

A bundle is a container that stores configuration files and Java classes used by the AOR utility. You must create a bundle in which to store new classes and packages you create for this demo application.

To create a new bundle:

1 Click **Main JCI Settings** in the AOR utility tree view.

2 Click **New** on the AOR utility toolbar.

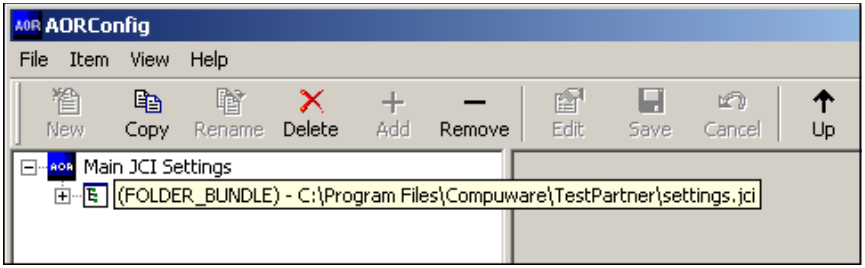
3 In the **Select a Bundle** dialog box, select **FOLDER_BUNDLE** and click **OK**. The **Browse For Folder** dialog box appears.

4 In the **Browse For Folder** dialog box, navigate to the installed location of the TestPartner application. The name of the installed location folder displays in the Folder field.

5 Place the cursor at the end of the installed folder name in the Folder field and type a backslash (\) and a name for the new folder bundle.

6 Click **OK**.

7 The left pane of the AOR Configuration utility displays the new item that has been added. The item shows the type of bundle created and its location.



You have just created a bundle in which to store the configurations. Now you will add packages to your bundle.

Building Support For a Control

When you identify the demo application in "Setting Up the Tutorial," the buttons are Java controls named **calcbutton** from the aordemo.calc package. In this section, you will use the AOR utility to add support for these buttons.

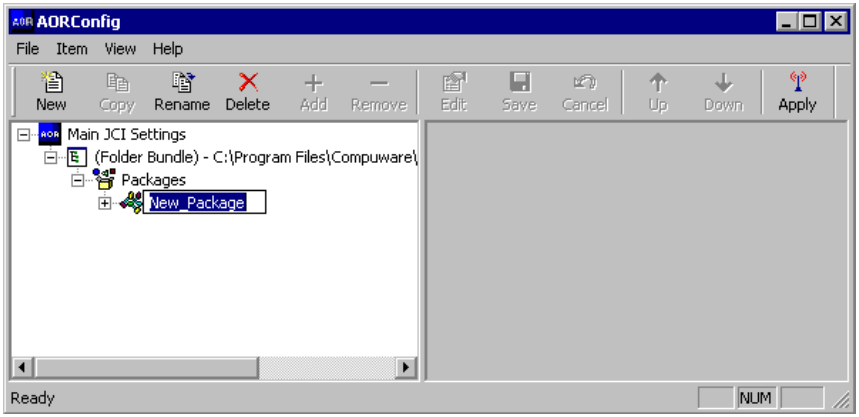
Note: To determine class and package names of controls in an application, consult the following: the application's developer, the API documentation of the application under test, or TestPartner's Property check. Remember, class and package names are case sensitive.

To add support for the aordemo.calc `package`.

1 In the AOR Configuration utility tree, expand the bundle you just created so that **Packages** and **Helpers** appear.

2 Select **Packages** under the bundle you just created.

3 Click **New** on the toolbar. A new package appears in the AOR utility tree with the temporary name **New_Package**.



4 Type the name **aordemo.calc** for the package.

Modifying General Component Settings

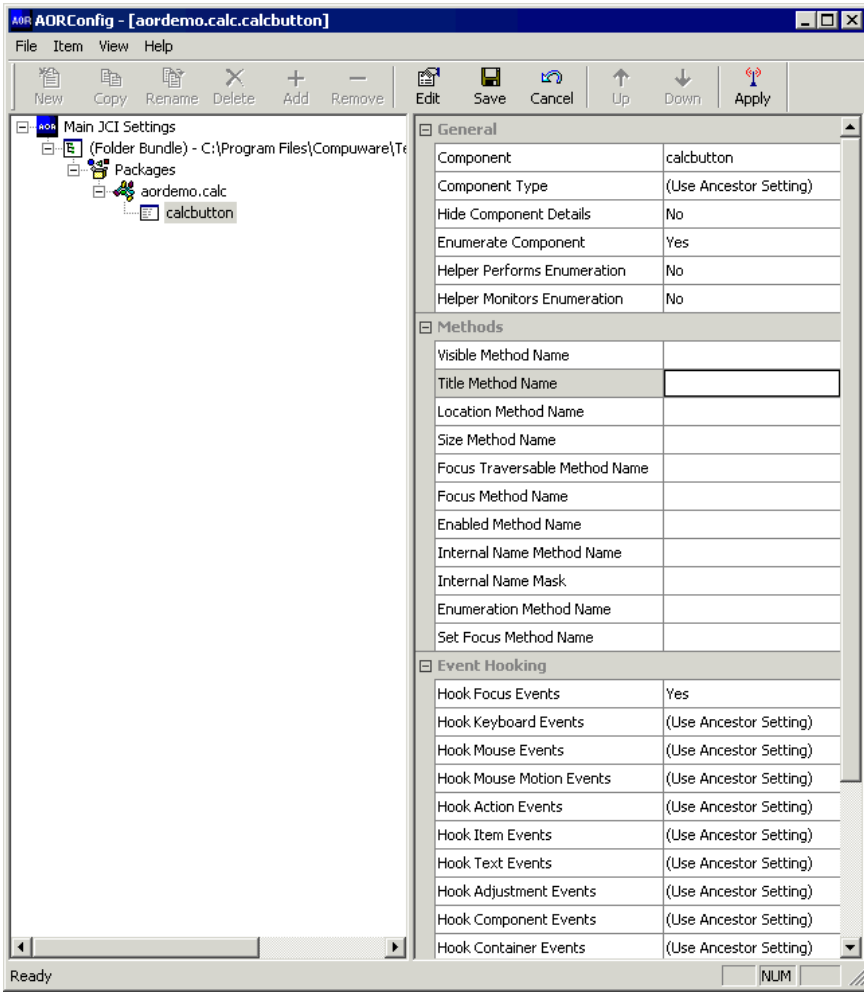
You can use the AOR utility to change the behavior of the testing tool when it detects calcbutton.

In this section, you modify support for the buttons so they appear as buttons to TestPartner. As you modify components, click **Save** on the toolbar.

1 In the AOR utility tree view, select the **aordemo.calc** package you just created and click **New**. A new component appears in the AOR utility tree view with the temporary name **New_Component**. Type the name **CalcButton** for the new component.

Note: Component names are case sensitive. Type the component name exactly as shown.

2 Double-click the **CalcButton** component. The **Component Configuration Properties** window appears in the right pane.



3 In the **General** section, click anywhere in the **Component Type** field.

4 Click the arrow to the right of the field and select **Button** from the list.

5 Click anywhere on the **Enumerate Component** field.

6 Click the arrow to the right of the field and select **No**.

Modifying Advanced Component Settings

Scroll down to the **Advanced** section and follow these steps:

1. Click anywhere on the **Record Options** field.
2. Click the arrow to the right of the field and select the following options:
   - ◇ Left Mousebutton Click
   - ◇ Left Mousebutton Doubleclick

   Clear all other check boxes.
3. Click anywhere on the **Playback Options** field.
4. Click the arrow to the right of the field and select **Testing Application**.
5. Click **Save** and **Apply** on the AOR utility toolbar.

Recording and Playing Back

Now that the testing tool knows that **CalcButtons** are **Buttons**, it can now identify, record, and play them back.

1. In TestPartner, create a new script.
2. Click **Scripts>Record** to begin recording the script.
3. In the demo application, click 9 x 3 =. The result should be 27.
4. Stop recording by double-clicking the TestPartner icon in the Windows taskbar.
5. Play back the script. The script should play back successfully.

The script recorded all the buttons by index, and not by the label that is on the button.

Recording the Button Text

Now add the ability to record each button by name.

1. Double-click the **CalcButton** component in the AOR utility tree, if it is not already open. The Component Configuration Properties Window appears in the right pane.
2. Scroll down to the **Methods** section. These configurations will override the methods that the testing tool uses to retrieve information about a particular control.

3 In the **Title Method Name** field, enter `getFaceText`. This is the method that the testing tool uses to retrieve the title/label of this control.

4 Click **Save** and **Apply** on the AOR utility toolbar.

5 Follow the steps in "Recording and Playing Back" on page 7. The buttons of the demo application calculator should now record and play back using the captions.

Building Advanced Support

The calculator application is now fully supported by the testing tool. In this section, you will build advanced support for this application using the table `aordemo.calc.CalcGrid`. This table contains the calculator's buttons.

Use the AOR utility to change the behavior of the testing tool when it detects the **CalcGrid**:

1 In the AOR utility tree view, select the package you created and click **New**.

2 A new component appears in the AOR utility tree with a temporary name. Enter the name `CalcGrid` for the new component.

3 Double-click **CalcGrid** on the AOR utility tree view. The Component Configuration Properties window appears in the right pane for the **CalcGrid** component.

4 In the **General** section, click anywhere in the **Component Type** field.

5 Click the arrow to the right of the field and select **Grid or Table.**

6 In the General section, click anywhere on the **Enumerate Component** field.

7 Click the arrow to the right of the field and select **Yes**.

8 Click anywhere on the **Helper Performs Enumeration** field.

9 Click the arrow to the right of the field and select **Yes**.

10 Click **Save** and **Apply** on the AOR utility toolbar.

11 Follow the steps in "Recording and Playing Back" on page 7. The buttons of the demo application calculator should now record and play back using the captions.

The testing tool should identify the main window, the calculator window, and the grid, instead of the buttons.

Creating a Helper Class

Helper classes are classes you create to work with the predefined classes to implement customized testing support. In this section, you will add support to record table select by position and by cell name. To accomplish this, you will create a helper class. A helper class is a standard Java class that extends **com.compuware.jci.HelperBase**.

Required: When a change is made to a helper file, you must remove and re-add the helper class to AOR utility, and then restart the Java application.

Create a helper that imports **com.compuware.jci** package and extends HelperBase (see AORHelperTemplate.java). At this point, no methods have been overridden. The enumeration, record, and playback functions for this control have not been changed. To provide new functionality or to change how the testing tool responds to this control, you must override one of the functions in the HelperBase. Each method is documented in the AOR utility online help topic "Developing Helper Classes".

Changing the Record Behavior

Begin by changing the record behavior of **CalcGrid**. You must override **getItemProperties** (see AORHelperLearn.java). This method receives the following two objects:

- CalcGrid object
- JCIProperties object

It returns the following boolean values:

- True for success
- False for failure.

This method is called by the testing tool during the record process to get more information about the control in question. TestPartner requires more information about what the user clicked on at a given location. It will set the x and y variable of the mouse click on that control in the JCIProperties object.

The **CalcGrid** component has associated methods called **getRow(y)** and **getColumn(x)** that convert the given x and y coordinates to a cell position within the table.

1. Use the x and y mouse positions to extract the row and column from the table.
2. Set the row and column properties in the JCIProperties object to the ones retrieved.

TestPartner uses this information to record a table by position.

Associating Text With the Cell

You should associate the text with the cell by calling the **getCell** method passing in the row and column. Then call the appropriate method to get the text from the component in that cell. For example, you could use the method **getFaceTex**t to get the text from the component **CalcButton**.

Earlier in the tutorial, you learned how to get the caption using the AOR utility. Now you must get the caption using Java code in the helper. To accomplish this, call the **getFaceText** method on the component retrieved previously. Use the **invokeMethod** method to call **getFaceText** using reflection.

Compile the code or use the pre-compiled class, `AORHelper-Learn.class`, included with the tutorial.

Add the helper to the AOR utility and the **CalcGrid** component:

1. In the AOR utility tree view, select **Helpers** under the bundle created earlier in "Creating a Bundle" on page 3.
2. Click **Add** on the AOR utility toolbar.
3. Select the file you just created, or the pre-compiled class, and click **Open**.
4. Select the **CalcGrid** component.
5. Click **Edit.** The **Component Configuration Properties** window appears in the right pane.
6. In the **Helper Class** field of the **Advanced** section, type the fully qualified class name of the helper you added.
7. Click anywhere on the **Record Options** field.
8. Click the arrow to the right of the field and select the following options:
   - ◇ Left Mousebutton Doubleclick
   - ◇ Left MouseButton Click
   - ◇ Use ItemProperties
9. Click **Save** and **Apply** on the AOR utility toolbar.

10 Open a new script and record the 9 x 3 operation as before. You should see TableSelect commands with the proper cell names. Use your testing tool to learn the table by position. Refer to your application's documentation for more information.

You will use this script for the rest of the tutorial.

Adding Support to Playback

In this section, you will add support to play back the script you just recorded using AORHelperReplay.java. First, override onReplay. This method receives and returns the same parameters as **getItemProperties** does in the above section.

The type of component you are working with will affect what the testing tool passes you in the JCIProperties object. For a Grid or Table, the testing tool passes a row and column to this method, or a cell name and column.

When the testing tool passes a cell name and column, search for the first match in the given column by obtaining the number of rows and cycling through each cell. Use the same process you did for overriding getItemProperties in "Changing the Record Behavior" on page 9. Once you have a match, get the cell bounds of the component in that cell, call a **doMouseClick** method in JCIReplayAgent, and compile the code or use **AORHelperReplay.class**, the pre-compiled class that is provided.

Change the `CalcGrid` component to use the new helper:

1 In the AOR utility tree view, select **Helpers** under the bundle created earlier in "Creating a Bundle" on page 3.

2 Click **Add** on the AOR utility toolbar.

3 Select the file you just created, or the pre-compiled class, and click **Open**.

4 Select the **CalcGrid** component.

5 Click **Edit**. The Component Configuration Properties Window appears in the right pane.

6 In the **Helper Class** field of the **Advanced** option, enter **AORHelperReplay**.

7 Click anywhere on the **Replay Options** field.

8 Click the arrow to the right of the field and select **JCI Helper**. Remember to deselect **Testing Application**.

9 Click **Save** and **Apply** on the AOR utility toolbar.

10 Play back the script from above.

Adding Support to Checks

Now add support for performing checks against tables. TestPartner will call the method **getContentProperties** in the helper class, passing the same parameters as in the previous two methods (see AORHelperCheck.java).

1. Obtain the number of rows and columns in the table.
2. Save these in the JCIProperties object as rows.count and columns.count.
3. Cycle through all the cells retrieving the text from each and set a property in the JCIProperties object using the following format:

   ```
   cell.row.column = cell text.
   ```

4. Compile the code or use the pre-compiled class, **AORHelperCheck.class**, included with the tutorial.

Add the Helper Class

Add the helper class to the AOR utility and change the CalcGrid component to use the new helper by following these steps:

1. In the AOR utility tree view, select **Helpers** under the bundle created earlier in "Creating a Bundle" on page 3.
2. Click **Add** on the AOR utility toolbar.
3. Select the file you just created, or the pre-compiled class, and click **Open**.
4. Select the **CalcGrid** component.
5. Click **Edit** and the Component Configuration Properties window appears in the right pane.
6. In the **Helper Class** field of the **Advanced** section, enter **AORHelperCheck.**
7. Click **Save** and **Apply** on the AOR utility toolbar.
8. Create a new content check in the testing tool. Click **Identify.**
9. Point to the table grid.

You have just created a content check to use within a script. You can use this check for verification of the table. For example, you can verify that the contents within that table remain the same.

Using the Enumerator Function

In this section, you will add support to retrieve the value from the Memory field (the green "M" on the demo application calculator). You must add the label that contains the setting back into the enumeration. Add a method to your helper class that overrides enumerate (see AORHelperFinal.java). This method receives the object under test and a JCIEnumerator object.

The JCIEnumerator object allows you to control the process of enumerating controls within a Java application. With this method you will check to ensure that the first component is a java.awt.Label and then add it to the enumeration by calling continueEnumeration passing in the java.awt.Label. This enables the internal support for java.awt.Label to handle the new control.

Compile the code or use the pre-compiled class, **AORHelperFinal.class**, included with the tutorial.

Follow these steps to add the new helper class to the AOR utility and change the CalcGrid component to use the new helper class:

1. In the AOR utility tree view, select **Helpers** under the bundle created earlier in "Creating a Bundle" on page 3.
2. Click **Add** on the AOR utility toolbar.
3. Select the file you just created, or the pre-compiled class, and click **Open**.
4. Select the **CalcGrid** component.
5. Click **Edit** and the **Component Configuration Properties** window appears in the right pane.
6. In the **Helper Class** field of the **Advanced** option, enter **AORHelperCheck.**
7. Click **Save** and **Apply** on the AOR utility toolbar.

You should see the Memory field during identification and in form checks.

When you record and play back the script, you should get the same result as you did from the script recorded previously. In this script however, the controls are treated as a table instead of a collection of buttons.

You have now completed the AOR utility tutorial and should have an understanding of the basic functions of the AOR utility. More information about the AOR utility is available in the AOR utility online help.