

Functional Test Tutorial Introduction

Time Required

To complete this tutorial, you will need approximately **2 hours**.

Learning Objectives

The Functional Test Tutorial is divided into 10 exercises that must be completed in sequence for the tutorial to work properly. The tutorial teaches you how to get started using Functional Test, and it walks through the major use cases for testing and how to perform basic operations. While completing the exercises, you will learn how to:

- Set up Functional Test to get started
- Create a Functional Test project
- Record a script
- Start your test application while recording
- Create verification points
- Play back a script
- View and understand the log
- Use the Comparator to update verification points
- Update the object map
- Change object recognition preferences
- Use regular expressions

Note: It is a good idea to print the tutorial before you begin and go through it from your printed copy for ease of use. You can click the link at the end of each exercise and print all the exercises before starting, by right-clicking inside each topic and clicking **Print**. You can also open the PDF file and print that.

When you are ready, begin Exercise 1.1: Setting up Functional Test

Exercise 1.1: Setting up Functional Test

IBM provides a Java Runtime Environment (JRE) that is installed and enabled for testing Java applications. You use that JRE for the tutorial. When you want to test your own Java or HTML applications, you must run the enabler and configure your environments and applications. For more information on these set-up tasks, see the Getting Started with Functional Test wizard on the First Steps page of the Welcome Page. For now you do not need to do anything to use the preconfigured JRE to continue.

Setting Logging Options

Functional Test provides several logging options. We'll use the HTML log.

1. To verify that this is the logging option you have set, click **Window > Preferences**.
2. In the left pane of the Preferences dialog box, expand **Functional Test**, then **Playback**, and click **Logging**.
3. Verify that the **Use Default** check box to the right of the **Log type** field is checked and **html** appears (grayed out) in the field.
4. Click **OK**.

This setting opens the HTML log automatically after you play back a script.

Creating a Functional Test Project

Before you start recording, you must create a Functional Test project.

1. In the Functional Test menu, click **File > New > Functional Test Project**.
2. Under **Project name**, type `FTtutorial`, without any spaces.
3. Under **Project location**, type `C:\FTproject`. Functional Test creates this directory.
4. If the source control option is available, do not select **Add the project to Source Control**.
5. If the associate project option is available, do not select **Associate the Functional Test Project with current Rational Project**.
6. Click **Finish**.

The FTtutorial project is now visible in the Functional Test Projects view, which is the left pane in the Functional Test perspective.


Now you are ready to begin Exercise 1.2: Recording a Script.

Exercise 1.2: Recording a Script



Before you begin, you must complete Exercise 1.1: Setting up Functional Test.

Begin recording



You are now ready to begin recording.

1. To start recording, click the **Record a Functional TestScript** button () in the Functional Test toolbar.
2. Select the FTtutorial project you just created, if necessary.
3. In the **Script name** field, type `Classics` (that is the name of the application you will be using).
4. Do not select the **Add script to Source Control** option if it is available.
5. Click **Finish**.


The Functional Test window automatically minimizes, and the Recording Monitor appears.

The Functional Test Recording Monitor appears every time you begin recording. You can minimize the monitor if you don't want it to be visible on the screen, and you can resize it. You can also click the **Display Toolbar Only** button (), which hides the monitor and shows only the toolbar. Click the **Display Monitor** button () to bring it back. Leave the monitor displayed during this tutorial.

The monitor displays messages for every action performed during your recording session, such as starting and pausing the recording, starting an application or browser, clicking and all other actions within an application, inserting verification points, and inserting other items into the script.

6. Click the **Monitor Message Preferences** toolbar button (). You can use these options to control the appearance of the text in the monitor any time.
7. Click **Cancel**.
8. Click the **Insert Script Support Commands** toolbar button (). This opens the Script Support Functions dialog box, which allows you to call another script, insert a log entry, insert a timer, insert a sleep command (a delay), or insert a comment into your script.
9. Click **Close**.

Starting the application

1. To start the test application, click the **Start Application** toolbar button ().
2. In the Start Application dialog box, use the arrow to select **ClassicsJavaA** , if necessary, and click **OK**.

The Functional Test Tutorial sample application, ClassicsCD, opens.

If the Recording Monitor is in front of the application, you can click the monitor's title bar and drag it to the lower right corner of the screen.

Recording Actions

You're going to record placing an order in this application.

1. Click the **+** next to **Haydn** to expand the folder in the composers tree.
2. In the list, click **Symphonies Nos. 94 & 98**.
3. Click the **Place Order** button.
4. In the Member Logon dialog box, keep the default settings of **Existing Customer** and **Trent Culpito**. Do not click either of the password fields at this time.
5. Click **OK**.
6. In the card number field, enter a credit card number. You must use the valid format of four sets of four digits here, for instance, 7777 7777 7777 7777.
7. In the expiration date field, enter a valid format expiration date, 07/07.

8. Click **Place Order**.
9. Click **OK** in the order confirmation message box.



Now you are ready to begin Exercise 1.3: Creating Verification Points.

Exercise 1.3: Creating Verification Points

Before you begin, you must complete Exercise 1.2: Recording a script .

In this exercise, you will record verification points to test objects. Verification points verify that a certain action has taken place, or verify the state of an object. You can create a Properties verification point or six types of Data verification points. When you create a verification point, you are capturing information about an object in the application to establish baseline information for comparison during playback. You will record a Data verification point to capture the tree of composers.

Creating a Data Verification Point


1. In the Recording Monitor, click the **Insert Verification Point or Action Command** button (.
2. On the **Select an Object** page of the Verification Point and Action Wizard, clear the advance to next page option if it is selected.
3. Use the Object Finder () to select the Composers tree in the application. Click the Object Finder and drag it over the tree. While holding down the mouse button, you will see that the entire tree is outlined with a red border and the object name is displayed (javax.swing.JTree) in a screen tip next to the red border. When you release the mouse button to make the selection, notice that the recognition properties for the object are listed in the grid at the bottom of the **Select an Object** page.
4. Click **Next**.
5. On the **Select an Action** page, **Perform Data Verification Point** should be selected. It is the first action on the page. Make sure it is selected and click **Next**.
6. On the **Insert Verification Point Data Command** page, in the **Data Value** field, select the **Tree Hierarchy** test. This test captures information about the entire tree hierarchy.
7. In the **Verification Point Name** field, type `Classics_tree` and click **Next**.
8. The **Verification Point Data** page displays the captured data in a grid in the right pane. If a check mark appears in the box beside an item, that item will be tested. By default, all items are selected. Leave them checked. If they are not selected, click the **Check All** button.
9. Click **Finish**.

Creating a Properties Verification Point

You can now insert a different verification point to confirm that the order is for the correct customer. A Properties verification point captures the text in the confirmation screen.

1. In the ClassicsCD application, click **Order > View Existing Order Status**. Do not click either of the password fields at this time.
2. Click **OK**.

You will test the label "Order for Trent Culpito" in the View Existing Orders dialog box.

3. In the Recording Monitor, click the **Insert Verification Point or Action Command** button (.
4. On the **Select an Object** page, select the advance to next page option this time (the check box beneath the Object Finder).
5. Drag the Object Finder over the label "Order for Trent Culpito" to select it. While holding down the mouse button, you'll see that the label is outlined with a red border and the object name is displayed (javax.swing.JLabel).

After you select the object, the **Select an Action** page opens because you selected the advance to next page option.

6. Select **Perform a Properties Verification Point**, which is the second action from the top.
7. Click **Next**.
8. On the **Insert Properties Verification Point Command** page, leave the **Include Children** field set to **None**.
9. Under **Verification Point Name**, accept the suggested default.
10. Leave the **Use standard properties** option checked, then click **Next**.

On the **Verification Point Data** page, the test object properties and their values are displayed in a grid format. You can choose which properties to test in the **Property** column and can edit the property values in the **Value** column.


By default, none of the properties are selected. To test object properties, you must choose the properties

you want to test by checking each of them. The properties you select are tested each time you play back a script with this verification point. You can check all properties in the list by clicking the **Check All** toolbar button above the grid. Use the **Uncheck All** button to clear all properties. When you use a Properties verification point, it is a good practice to test only the properties you are interested in.

In this case, only the **text** property is of interest to determine whether the order is for the correct customer.

11. In the **Property** column, scroll to the **text** property. Select the box next to the **text** property to test it during playback. You may have to click twice in the check box for the check mark to persist.
12. Also, select the **opaque and visible** properties.
13. Click **Finish**.
14. In the ClassicsCD View Existing Orders dialog box, click **Close**.

Now let's do another quick order to test the password fields we did not test earlier.

1. Expand the **Schubert** folder in the composers tree.
2. Click **String Quartets Nos. 4 & 14**.
3. Click **Place Order**.
4. In the Member Logon dialog box, keep the default settings of **Existing Customer** and **Trent Culpito**.
5. This time, enter xxxx in the **Password** field.
6. Click the check box to select the **Remember Password** option.
7. Click **OK**.
8. Enter a valid format credit card number and expiration date, for instance, 7777 7777 7777 7777, expiration 07/07.
9. Click **Place Order**.
10. Click **OK** in the order confirmation message box.
11. Close the ClassicsCD application by clicking the **x** button.
12. Click the **Stop Recording** button () on the **Recording** toolbar.


When you stop recording, Functional Test closes the monitor, and writes your script and object map to your project directory. The Functional Test window is restored and the script is displayed in the main window.

Now you are ready to begin Exercise 1.4: Playing Back Scripts.

Exercise 1.4: Playing Back Scripts

Before you begin, you must complete Exercise 1.3: Creating Verification Points.

After we play back the script we'll come back to the Functional Test window and look at the different sections. Because the script you just recorded is the active script (the tab in the Java Editor should be `Classics.java`), that script will play back when you click the playback button.

1. To play back the script, click the **Run Functional Test Script** button () on the Functional Test toolbar, or click **Script > Run**.
2. In the Select Log dialog box, keep the default log name of **Classics** and click **Finish**.

Functional Test is minimized, and the Playback Monitor starts at the top right of your screen. As the script plays back, messages are displayed in the Playback Monitor. Functional Test plays back all of your recorded actions, such as the application starting, the actions you performed on the application, and the verification points.

When playback is finished, the HTML log displays the results of the run in a separate window. All events listed in the log should include **Pass** in the event headings in green. Notice that the two verification points you recorded are listed.

3. **Close** the log.

Now that you have successfully recorded a script and played it back, let's look at the Functional Test perspective in more detail.

1. If the Functional Test window is minimized, restore it.

When you have multiple scripts, Functional Test displays all open scripts in a project in the Java Editor (the script window). Each script has a tab in the banner of the window, and the tab for the active script is in color. Any actions you perform on a script, such as playing it back, are performed on the active script.

Throughout the script, notice the information about the script shown at the top in light blue and prefixed by asterisks. This information comes from the script template, which you can modify. For more information about modifying the script template, see the Functional Test Help.

Notice also that Functional Test adds a short comment to the script in green characters to identify the object that the following lines refer to. This information makes it easier to navigate the script. Strings passed as arguments to methods during recording, including user inputs, are bright blue.

When your cursor hovers over certain areas of the script, Functional Test displays useful information in a pop-up text box. For example, for a helper method, you see the description property set in the object map followed by the recognition properties of the object. The hover feature is controlled by Preferences. To turn it off or modify what is shown, click **Window > Preferences**, then choose **Editor** and click the **Hovers** tab. The hover feature is on by default.

To the left of the Java Editor (the script window) is the Functional Test Projects view, which lists any Functional Test projects to which you are currently connected. All scripts within each project are listed below the project name. This Projects view provides another way to navigate to a different script. When you double-click a script in the Projects view, it opens in the script window and becomes the active script.

To the right of the Java Editor is the Script Explorer, which lists the verification points and object map of the active script. From the Script Explorer, you can start the Verification Point Editor to display and edit verification points, and start the object map editor to display and edit object maps. For more

information about the Script Explorer or the other parts of the Functional Test perspective, such as the Tasks View and Console View, see the Functional Test Help.

Now you are ready to begin Exercise 1.5: Looking at Verification Points and Object Maps.

Exercise 1.5: Looking at Verification Points and Object Maps

Before you begin, you must complete Exercise 1.4: Playing back scripts.

Looking at verification points

You can examine and modify the data inside a verification point.

1. In Functional Test, verify that your script, `Classics.java`, is still the active script in the Java Editor.
2. The two verification points you recorded should be listed in the Script Explorer to the right of the script. If necessary, click the plus sign (+) next to **Verification Points** to expand them.
3. Double-click **Classics_tree**.

This is the first verification point you recorded, on the list of composers. The Verification Point Editor starts; you can update verification point data for future playbacks.

The Data verification point has six possible display types. This is a Data (tree) verification point. The object type is a tree, in this case, a `javax.swing.JTree`. To edit the data in this tree, double-click any of the sub-items in the tree to open a small edit box where you can make changes. Use the check boxes beside each item to indicate whether you want this item to be tested in future playbacks. Checked items get tested. To learn more about using the Verification Point Editor, see the Functional Test Help.

4. Close the Verification Point Editor.

Looking at object maps

You can also examine and modify the data inside the object map.

1. In the Script Explorer, expand the **Test Objects** folder.

The first item, **Private Test Object Map**, is the script's object map. The individual objects listed beneath Private Test Object Map are references to objects that were acted on during recording.

2. Double-click **Private Test Object Map** () to open it .

When you record a script, Functional Test creates an object map for the application under test. Each script is associated with an object map file. The map file can be private -- associated with only one script -- or shared among many scripts. When you recorded the script, Functional Test used the default setting (private map) on the second page of the Record a New Functional Test Script dialog box. The object map contains properties for each object, and you can easily update the information in one central location. Then, any scripts that reference that object also share the updated information.

In your object map, the top-level object of a frame, lists all the objects the frame includes beneath the frame object.

3. Expand the top-level object that is called "Java: Frame: logFrame1: javax.swing.JFrame."

The frame object includes the logon dialog box. The radio buttons, password fields, and action button are listed beneath the frame object.

4. Click one of the objects.

Notice that the recognition properties appear in the grid below the object tree.

The object map also provides a quick way to add object references to a script. In the object map menu, you can click **Test Object > Insert Object(s)** to add objects. For more details on adding objects, see the Functional test Help.

You can also perform other operations from the object map, such as changing the weight of a recognition property and editing recognition properties and values. We'll do several advanced procedures using the object map later on.

5. In the object map menu, click **Preferences > Clear State On Close**.

The **Clear State On Close** command is a toggle menu item and should be on by default, so you will be clearing it. If it were left on, when you close the map all objects would be accepted. We want to do that in a later step when we return to the object map to make changes.

6. Close the object map. Do not save any changes you may have made.

Now you are ready to begin Exercise 1.6: Regression Testing.

Exercise 1.6: Regression Testing

Before you begin, you must complete Exercise 1.5: Looking at Verification Points and Object Maps.

When you have a new build of the application, you can run the automated test you recorded by playing back your script on the new build. To execute your script on the new build, you must change the name of the application in your script. (You would not need to do this on a development project; you do it here to simulate getting a new build of the application.)


1. In the Java Editor (script window), verify that your script (Classics.java) is the active script.

At the top of the script, beneath the template information, is the start application command:

```
startApp("ClassicsJavaA");
```

2. Change the "A" to "B".

Java code is case-sensitive, and so be sure to use an uppercase B. You do not need to save or compile the script for the change to take effect. It is done automatically when you run the script.

3. Click the **Run Functional Test Script** toolbar button () to play back the script.
4. In the Select Log dialog box, select **Classics** if necessary and click **Finish**. You will be prompted to overwrite the log.
5. Click **Yes**.

The script begins to play back quickly, but slows near the end on the Member Logon dialog box. That is because Build B of the application has different text in the field beside the check box. Functional Test is looking for an object that matches the recognition properties recorded in Build A. We'll show how to fix this problem later on.

6. When the log opens after playback, look at the messages.

You should see one failure and one warning in the log. The second verification point -- the one on the label -- failed because of a change in the application. Next, we'll see how to update the verification point baseline to fix this. An object recognition warning was generated for the password check box field. We'll also show how to fix that in the object map using a regular expression in a later section of the tutorial.

Did you notice that the main screen of ClassicsB looks different from ClassicsA, but that did not cause the script to fail? The same objects are present but in a different location on the two applications. This did not cause a failure because Functional Test uses robust recognition methods to locate the objects. For example, it does not rely on superficial properties such as screen coordinates to find objects. Instead, it uses internal recognition properties. This method allows for flexibility in the user interface design, without requiring that you alter or re-record your scripts.

7. Leave the log open.

Now you are ready to begin Exercise 1.7: Using the Comparator to Update a Verification Point.

Exercise 1.7: Using the Comparator to Update a Verification Point

Before you begin, you must complete Exercise 1.6: Regression testing.

Verification points provide a baseline of the properties or data of an object. If the verification point fails on a subsequent build of an application, you have found a defect or an intentional change to the application. If the change is intentional, you can update the information in the verification point so that the test continues to be valid for future builds.

1. If you closed the log, reopen it by double-clicking on the log name in the Projects view.
2. In the log, click the **View Results** link at the end of the failed verification point entry. The event heading is "Verification Point (OrderforTrentCulpito_standard)."

The Functional Test Verification Point Comparator displays your verification point data. Notice that the Comparator banner includes the name of your verification point. (Note: If you get an error message and the Comparator does not open, you need to enable the Java plug-in of your browser. For the instructions to do that, see the topic called "Enabling the Java Plug-in of a Browser" in the "Before You Record" section of the Functional Test Help. Use the **Help** menu in Functional Test to access this section in the Help table of contents.)


When a verification point fails, the Comparator shows the expected and the actual values, to help you analyze the differences. You can then load the baseline file and edit it or update it with the values from the actual file.

Failures are displayed in red.


3. Scroll to the **text** property.

When you created the verification point on ClassicsA, the banner title was "Order for Trent Culpito." When you played back the script on ClassicsB, the banner title was "Orders for Trent Culpito." "Orders" is correct, because a customer might have multiple orders in the Orders dialog box. So you must update the baseline file to change the text to match ClassicsB.


You can edit only the baseline file.

4. Click the **Load Baseline to Edit** button () on the Comparator toolbar.

Notice that the left **Value** column displays the **Baseline Value** now.

5. Instead of scrolling to the **text** property, you can click the **Jump to First Difference** button () above the **Property** column. The four navigation buttons help you locate the differences between the baseline and actual files.

You can update the baseline file in two ways. You can edit that cell of the grid, adding the letter s to the word "Order," or you can use the **Replace Baseline** command. Replacing the baseline replaces all values from the baseline file with the values from the actual file. In general, if you need to edit only one or a few values, you should edit the individual values.

6. This test has only one difference to update, so click the **Replace Baseline with actual value** button () on the Comparator toolbar.

Both values in the **text** property now match and the property no longer appears in red. For more information about using the Comparator, see the Functional Test Help.

7. **Close** the Comparator.

Let's play back the script again to confirm the verification point passes, now that you updated the baseline value for the failure.

8. Close the log first.
9. Click the **Run Functional Test Script** button on the Functional Test toolbar.
10. Select the **Classics** log and click **Finish**.
11. Click **Yes** if prompted to overwrite the log.

Functional Test pauses on the Member Logon dialog box, since you did not fix that recognition problem yet. At the end of playback, Functional Test displays the log. The verification point now passes! See how easy it is to use the Comparator to update object data and properties to account for changes in the application under test.

12. Leave the log open.

Now you are ready to begin Exercise 1.8: Updating the Object Map.

Exercise 1.8: Updating the Object Map

Before you begin, you must complete Exercise 1.7: Using the Comparator to update a Verification Point.

In this exercise, you will fix the object recognition warning by using the object map. You will also use a regular expression for more flexible object recognition.

When you see a recognition failure or warning, look at the log message. At the end of Exercise 1.7, you should have kept the log open. If it is not open, open it by double-clicking the log in the Projects view. One individual warning remains in the log. The event heading is "Object Recognition is weak (above the warning threshold)."

1. Look at the **ObjectLookedFor** and **objectFound** fields in the warning section near the bottom of the log.

In ClassicsA, the name of the password field is **Remember Password**. In ClassicsB it is **Remember The Password**. When you played back the script on ClassicsB, the object recognition did not match exactly because of this difference.

2. Look at the **Line Number** field in the log and note the number.
3. Close the log and return to Functional Test.
4. Click anywhere in the script window, and then click **Navigate > Go to Line**.
5. Type the line number from the log failure message, and click **OK**.


The cursor moves to the left margin of that line number. You can also find the line number by looking at the indicator in the bottom of the Functional Test window. The line number and cursor position within the line is shown there. For example, "43:9" refers to the position on line 43 that is 9 characters to the right of the left margin. The line in your script should be:

```
RememberPassword().clickToState(SELECTED);
```

This line represents your click on the password check box. This line in the script shows which object is failing. Now you can look for that object in the object map.

6. To find the object, return to the list of Test Objects in the Script Explorer (right pane).

You should see "rememberPassword" listed under the **Test Objects** folder.

7. Double-click the rememberPassword object to open it in the object map.
8. Click **Test Object > Accept All**  on the object map menu. If the command is grayed out, don't do anything.

Notice that all the objects change to black text. The text is blue (to indicate new objects) until you accept the objects in a map. You should accept the objects the first time you look at a newly created object map.

9. If the password check box object is not selected in the map, select it. (It is the object called "Java: checkBox: checkRemember: javax.swing.JCheckBox.")
10. Look at the recognition properties listed in the **Recognition** tab at the bottom of the object map.

You can see that this is the object from ClassicsA, because it says "Remember Password" in the **accessibleContext.accessibleName** property. This is the "old" object. However, when you played back the script on ClassicsB, the text for that object changed, so Functional Test recognizes it as a "new" object. You want to use the new object properties in this case, so you must add it to the map.

To add the new object to the map, open ClassicsB and the Member Logon dialog box.

11. Click **Applications > Run** on the object map menu.
12. Use the arrow to select ClassicsJavaB. (Be sure to pick B.)
13. Click **OK**.
14. In ClassicsCD, select any CD and click **Place Order**.

The Member Logon dialog box opens.

15. Move the object map lower on your screen, if necessary, to see all of it. In the object map menu, click **Test Object > Insert Object(s)**.

This is the same as the Object Finder tool in the **Select an Object** page of the Verification Point Wizard.

16. Clear the advance to next page option if it is selected.
17. Use the Object Finder tool to select the Remember the Password check box in the Member Logon dialog box.

After you select the check box, you'll see that the **accessibleContext.accessibleName** property is now "Remember The Password ." Stretch the borders of the object map, if necessary, to see the properties.

18. On the **Select an Object** page, click **Next**.
19. Don't change anything on the **Select Object Options** page, and click **Finish**.

The new check box object is now shown in the object map.

20. Click another object and notice that the new item is listed in blue and the word "New" appears at the beginning of the line.

Now both the old and the new objects are listed in the map. You want to unify the two objects and take the properties from each that you want for the new object.

21. To unify the objects, click the old object (the original check box labeled "CheckBox: checkRemember"), and drag it onto the new object in the list. Position the tip of the cursor arrow over the new object before you release the mouse button.

The Unify Test Objects wizard appears.

22. Widen the Unify wizard if necessary to see more of the information in the lower sections.

In the lower left section, the original object's properties are shown. It should be labeled "Source: RememberPassword." That is what the text was on the check box in ClassicsA. In the lower right section, it should be labeled "Target: RememberThePassword." That is what the text is on the check box in ClassicsB.

Because you dragged the old object to the new object, the new object's recognition properties are filled in at the top of the wizard. In general, Functional Test puts the new properties at the top if they are the preferred properties. However, some old administrative properties might be preferred. For example, Functional Test retains regular expressions in the old property set. To use a property from the old object, double-click that property in the grid of the old object and it will be copied up into the unified object. In this case, we want to use all the properties of the new object, which are already filled in.

23. Click **Next**.

All scripts that are affected by this change in the object map are listed. Only one script, Classics, is affected.

24. Click **Finish**.
25. In the object map, click the **File: Save** button on the object map toolbar to save the changes you made, and close the object map.

Now we'll play back the script again on ClassicsB to confirm that it passes.

1. First close both dialog boxes of ClassicsCD.
2. In Functional Test, click **Run Functional Test Script** on the toolbar.
3. Select the **Classics log** and click **Finish**.

The script now passes with no warnings! Notice that the playback no longer pauses on the password check box object because the recognition properties now match.

This object unification feature is an easy way to update scripts when recognition properties of an object intentionally change. One of the major advantages of this feature is that if your object map is being used by many scripts, you could update them all when you make the change in the wizard. Instead of manually editing multiple scripts, you can make a change once in the map and the change propagates automatically to all scripts that use it. This feature can save you time.

Note: There is also an easier way to update the recognition properties of a test object should they change. Instead of using the Unify wizard as described in this exercise, from the Object Map you can select the test object whose recognition properties you want to update. Right-click on the test object as it is displayed in the Object Map tree and select **Update Recognition Properties** from the pop-up menu. You will need to have the test application running when this action is performed so that Functional Test can get the updated recognition properties. You would only use this update method if you do not want to use any properties of the old object.

4. Close the log.

Now you are ready to begin Exercise 1.9: Changing Recognition Preferences.

Exercise 1.9: Changing Recognition Preferences

Before you begin, you must complete Exercise 1.8: Updating the Object Map.

In the previous exercise, you saw how you can update the recognition properties of an object when they change. Another factor you can change is the recognition weighting that Functional Test uses during playback. You use the ScriptAssure recognition preferences to set this. The label object that you tested with the second verification point can demonstrate how this works.

1. On the Functional Test menu, click **Window > Preferences**.
2. Click **Functional Test > Playback > ScriptAssure**.
3. Click the **Advanced** button.

Notice that one of the default settings is **Warn if accepted score is greater than: 10000**. A score of 10000 indicates that one important property can be wrong. Let's lower the score to 5000 and see what happens.

4. Click the **Use Default** check box beside this field.
5. Then type 5000 in the field. Click **OK**.
6. Play back the script on ClassicsB again.

The log now contains a warning for the label object. The reason given in the **objectFound** field, is that the recognition score is 10000. This discrepancy was caused by changing the word "Order" to "Orders" in the label.

7. Close the log.

Restore the default value for the recognition score.

1. Click **Window > Preferences**.
2. Click **Functional Test > Playback > ScriptAssure**.
3. Click the **Advanced** button.
4. Click the **Use Default** check box beside the **Warn if accepted score...** field.

This will change the 5000 back to 10000.

5. Click **OK**.
6. Play back the script again.

Now the warning is gone, and everything passes.

7. Close the log.

This exercise showed how you can tweak the recognition score in order to achieve the sensitivity that you want for object recognition. For more information about using ScriptAssure, see the Functional test Help.

Now you are ready to begin Exercise 1.10: Using Regular Expressions.

Exercise 1.10: Using Regular Expressions

Before you begin, you must complete Exercise 1.9: Changing Recognition Preferences.

The last thing we'll do using the object map is convert a property value to a regular expression. In this case, the regular expression allows for more flexibility in the object recognition.

We just saw how the script passes completely on ClassicsB now. That was our goal because the changes made to the application in ClassicsB are correct. So the script is now in the state we want it to be in going forward. Now when you play the script back against ClassicsA, it fails because of the changes made earlier. You might want to allow more than one variant of an object to pass. You might have a dynamic object or have several versions of your application with slightly different versions of an object, in which both are correct. You can use a regular expression to allow more than one version of a property value, such as text, to accommodate this scenario.

1. To play back against ClassicsA, edit the startApp command at the top of the script and change the B to an A.
2. Click **Run Functional Test Script** on the Functional Test toolbar.

During playback, Functional Test pauses a little on the password check box object, but eventually it finishes.

The script now gives a warning. Notice in the log that it's the same object, the password check box.

3. Close the log.
4. Open the object map from the password check box object as you did in Exercise 1.8, by double-clicking on the password check box in the Script Explorer.
5. Open the application by clicking **Applications > Run** in the object map.
6. Use the arrow to select ClassicsJavaA. (Make sure to pick A.)
7. Click **OK**.
8. Pick any CD and click **Place Order** in ClassicsCD to open the Member Logon dialog box.
9. Add the new object into the map as you did in Exercise 1.8, by clicking **Test Object > Insert Object (s)**, and use the Object Finder to select the password check box in the Member Logon dialog box in the application.
10. Click **Next**, and then click **Finish**.
11. In the top pane of the object map, drag the old check box object to the new check box object to unify the objects.
12. Widen the Unify Test Objects wizard by dragging one of the sides outward to make the fields longer, if necessary.

You will use two different regular expressions: one on the **name** property and one on the **accessibleName** property.

The unified object is shown in the **Unified Test Object Properties** grid (top pane); the **name** property has a value of "checkRemember."

13. In the top pane, right-click the "checkRemember" value and click **Convert Value to Regular Expression**.

Functional Test designates the value as a regular expression by the "xy" icon in front of the value text.

14. Double-click the **name** value again so that you can edit the field.
15. Delete the word "check," and edit the remainder to read:

```
[rR]emember
```

16. Click outside of that cell.

This pattern allows the word "remember" with either an uppercase "R" or lowercase "r" to pass. This is important because the comparisons are case-sensitive, and only an exact match will pass.

The value of the **accessibleName** property is "Remember Password."

17. Right-click the Remember Password value and select **Convert Value to Regular Expression** to convert it.
18. Then double-click the value and edit it to read:

```
Remember.*Password
```

You are removing the space and adding the period (.) and asterisk (*) characters.

19. Click another cell.

The "." allows any character to appear in that position. In one version of the application, there is a space between the two words in this property, and in the other version there is no space. This pattern covers both cases.

20. Click **Next**, and then click **Finish** in the wizard.
21. Click **File: Save** in the object map to save the changes, and close the object map.
22. Close ClassicsCD.

Now here comes the fun part!

23. Play back the script again on ClassicsA.

The object recognition warning on ClassicsA no longer appears in the log.

24. Close the log.
25. Change the startApp command to play back ClassicsB, and run the script.

The object recognition also passes on ClassicsB! Regular expressions offer more flexible recognition for an object that has different properties in different versions of an application, and both are recognized during playback. For more information about regular expressions, see the Functional Test Help.

Finish your tutorial by reviewing the materials in the Summary.

Perform Functional Test Basic Operations Summary

This Functional Test Tutorial has shown you how to set up Functional Test for testing, record and play back scripts, create verification points and use the Verification Point Comparator to update object properties or data, and several ways to use the object map to your advantage. There are many other features and use cases of Functional Test, such as the ClearCase integration and cross-platform playback. Look through the table of contents in the Functional Test Help for more information about using Functional Test.

Completed Learning Objectives

If you have completed all of the exercises, you should now be able to ...

- Create a Functional Test project
- Record a script against actions on your test application
- Start your test application properly while recording
- Create verification points
- Play back scripts
- Use the Functional Test log
- Update verification points using the Comparator
- Update the object map
- Change recognition preferences for an object
- Use regular expressions for more flexibility in object recognition

More information

If you want to learn more about the topics covered in this tutorial, consider the following sources:

- [Functional Test Help](#)
- [Functional test API Reference](#)
- [Functional Test Welcome Page](#)