Rational®

IBM

Create a functional test

Contents

Create a functional test

This Rational® Functional Tester tutorial walks you through the major use cases for creating and playing back functional tests. This comprehensive tutorial uses a sample Java™ application that is installed with the product.

Learning objectives

After completing this tutorial, you will be able to do the following:

- Create a functional test project and record a script
- Work with verification points, object maps, and regular expressions
- Use a comparator to update a verification point
- Play back a script
- Perform regression tests

Time required

45 minutes

Related information

 View the PDF version

Tutorial: Create a data-driven functional test

Tutorial: Automate a manual test that is based on keywords

Sample: Functional test project

Introduction: Create a functional test

This tutorial teaches you how to get started using Functional Tester and walks you through the major use cases for testing and performing basic operations. This tutorial uses the sample application provided with Functional Tester to perform all the tasks.

The Functional Tester tutorial is divided into 10 lessons that must be completed in sequence for the tutorial to work properly.

Learning objectives

After completing this tutorial, you will be able to:

- Create a functional test project and record a script
- Work with verification points, object maps, and regular expressions
- Use a comparator to update a verification point
- Play back a script
- Perform regression tests

Note: Consider printing the tutorial before you begin and using the printed copy as you work through the lessons. You can either print the PDF version of the tutorial or print each individual lesson by right-clicking inside each topic and clicking **Print**.

Time required

This tutorial should take approximately 45 minutes to finish. If you explore other concepts related to this tutorial, it could take longer to complete.

Prerequisites

This is an introductory tutorial. You should be able to perform the tasks with little or no experience with Rational Functional Tester.

Lesson 1: Set up Rational Functional Tester

IBM® provides a Java Runtime Environment (JRE) that is installed and enabled for testing Java applications. Use this 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 Tester wizard in the First Steps section of the product Welcome. For now you do not need to do anything to use the preconfigured JRE to continue.

Start Rational Functional Tester and then perform the following tasks before you record your first test script.

Set logging options

Rational Functional Tester provides several logging options. We will use the HTML log.

1. Click **Windows** → **Open Perspective** → **Other** to open the functional test perspective. In the Open Perspective dialog box, select the **Functional Test** option.
2. To verify that HTML logging is set, click **Window** → **Preferences**.
3. In the left pane of the Preferences window, expand **Functional Test**, then **Playback**, and click **Logging**.
4. Verify that the **Use Default** check box is selected and that **html** appears in the **Log type** field.
5. Click **OK**.

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

Create a Functional Tester project

Before you can start recording, you must create a Functional Tester project.

1. In the Functional Tester menu, click **File** → **New** → **Functional Test Project**.
2. Under **Project name**, type `FTtutorial` (no spaces).
3. Under **Project location**, type `C:\FTproject`. Functional Tester 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.

Lesson 2: Record a script

In this lesson, you will record a script using the Functional Tester Recording Monitor.

Begin recording

You are now ready to begin recording.

1. To start recording, click the **Record a Functional Test Script** button ( ) in the Functional Test toolbar.
2. Select the FTtutorial project that you just created.
3. In the **Script name** field, type `Classics` (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 Tester window automatically minimizes, and the Recording Monitor is displayed.

   Learn more about Recording Monitor: The Functional Tester Recording Monitor is displayed 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 also resize it. You can also click the **Display Toolbar Only** button ( ), which hides the recording 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 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 any time to control the appearance of the text in the monitor.
7. Click **Cancel**.
8. Click the **Insert Script Support Commands** toolbar button ( ).

   This opens the Script Support Functions window, 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**.

Start the application

1. To start the test application, click the **Start Application** toolbar button ( ).
2. In the Start Application window, select **ClassicsJavaA** and then click **OK**.

   The Functional Tester Tutorial sample application, ClassicsCD, opens. If the Recording Monitor is in front of the application, you can drag it to the lower right corner of the screen.

Record actions

You are 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 window, 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 window.

Lesson 3: Create verification points

In this lesson, 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, Image verification point or six types of Data verification points. When you create a verification point, you capture information about an object in the application to establish baseline information for comparison during playback.

Create a data verification point

You will record a Data verification point to capture the tree of composers.

1. In the Recording Monitor, click the **Insert Verification Point or Action Command** button ( ).
2. In the Select an Object page of the Verification Point and Action Wizard, clear the **After selecting an object 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. In the Select an Action page, make sure **Perform Data Verification Point** is selected and click **Next**.
6. In 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**.

Create an image verification point

You can insert an image verification point to confirm that the appropriate album is displayed for the selected CD.

1. In the Recording Monitor, click the **Insert Verification Point or Action Command** button ( ).
2. In the Select an Object page of the Verification Point and Action Wizard, clear the **After selecting an object advance to next page** option if it is selected.
3. Use the Object Finder ( ) to select the Album image in the application. Click the **Object Finder** and drag it over the album image. While holding down the mouse button, you will see that the album image outlined with a red border and the object name is displayed (`javax.swing.JLabel`) 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. In the Select an Action page, select**Perform Image Verification Point** and click **Next**.
6. In the Insert Image Verification Point Command page, type `Album_image` as the **Verification Point Name**.
7. Make sure that the option **Select full image** is selected and click **Next**.
8. The Verification Point Data page displays the captured image in the right pane. Click **Finish**.

Create 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 window.
3. In the Recording Monitor, click the **Insert Verification Point or Action Command** button ( ).

4. In the Select an Object page, select the **After selecting an object advance to next page** option.
5. Drag the **Object Finder** over the label "Order for Trent Culpito" to select it. While holding down the mouse button, note 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, and then click **Next**.
7. On the Insert Properties Verification Point Command page, confirm that the **Include Children** field is set to **None**.
8. Under **Verification Point Name**, accept the suggested default.
9. Leave the **Use standard properties option** selected and 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.

   Learn more about selecting object properties: By default, none of the properties are selected. To test object properties, choose the properties that you want to test by selecting each property. The properties you select are tested each time you play back a script with this verification point. You can select all properties in the list by clicking the **Check All** toolbar button above the grid. Use the **Uncheck All** button to clear all properties. For best results when using a Properties verification point, 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.
10. In the Property column select the **text**, **opaque**, and **visible** properties to test them during playback. You may have to click the check box twice for the selection to persist.
11. Click **Finish**.
12. In the ClassicsCD View Existing Orders window, click **Close**.

Test the password fields

Now let us place another quick order to test the password fields that we did not test earlier.

1. Expand the **Haydn** folder in the composers tree.
2. Click **Symphonies Nos. 94 & 98**.
3. Click the **Place Order** button.
4. In the Member Logon window, keep the default settings of **Existing Customer** and **Trent Culpito**.
5. This time, type `xxxx` in the **Password** field.
6. Select the **Remember Password** option.
7. Click **OK**.
8. Type a valid **card number** 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, Rational Functional Tester closes the recording monitor and then writes your script and object map to your project directory. The Rational Functional Test window is restored and the script is displayed in the main window.

Lesson 4: Play back the script

In this lesson, you will play back the script and look at some parts of the Rational Functional Tester interface. Because the script you just recorded is the active script, 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.
2. In the Select Log window, keep the default log name **Classics** and then click **Finish**.

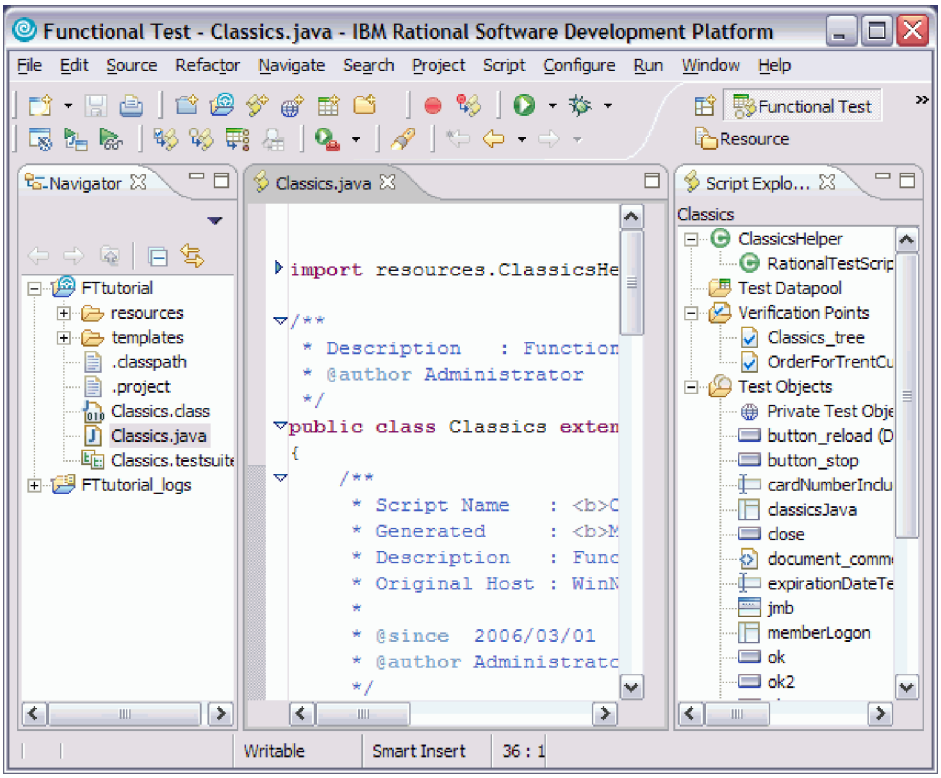
   Rational Functional Tester is minimized, and the Playback Monitor starts at the top right of your screen. As the script plays back, messages are displayed in the Playback Monitor. Rational Functional Tester plays back all of your recorded actions, such as the application starting, the actions you performed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

   When playback is finished, the HTML log displays the results of the test run in a separate window. Each event listed in the log should include Pass in the event headings in green. Notice that the two verification points that you recorded are listed.
3. Close the log. Now that you have successfully recorded a script and played it back, let us look at the Functional Test perspective in more detail.
4. If the Functional Test window is minimized, restore it. When you have multiple scripts, Functional Tester displays all open scripts in a project in the Java Editor (the script window).

   Learn more about Java Editor: 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 Tester Help.

   Notice that Functional Tester 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 Tester 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 **Java** → **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 Tester 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 Tester Help.

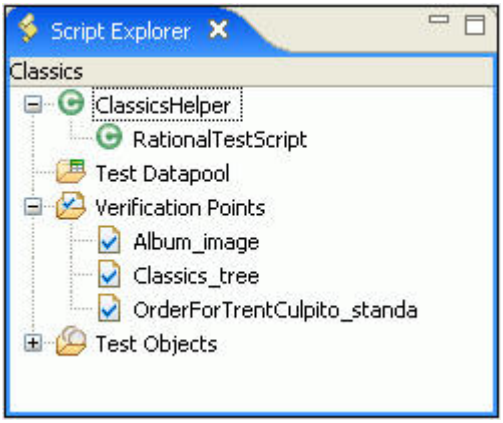
Lesson 5: View verification points and object maps

In this lesson, you will learn how to view and modify the properties of verification points and object maps.

View verification points

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

1. In Rational Functional Tester, verify that your script, Classics.java, is still the active script in the Java Editor.
2. The three 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 the list.



3. Double-click **Classics_tree**.

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

   Updating verification points: Data verification points have 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. To learn more about using the Verification Point Editor, see the Functional Tester Help.

4. Close the Verification Point Editor.

View object maps

You can 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 object map for this script. The individual objects listed under Private Test Object Map are references to objects that were acted on during recording.

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

   Object map types: When you record a script, Functional Tester 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 one script exclusively -- or shared among many scripts. When you recorded the script, Rational Functional Tester used the default setting (private map). 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.

3. Expand the top-level object `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 are displayed 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. 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 perform several advanced procedures using the object map later in the tutorial.

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, all objects would be accepted when you close the map. 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.

Lesson 6: Perform regression tests

In this lesson, you will execute your script on a different build. When you have a new build of an 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, note 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 window, select **Classics** and then 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 window. That is because Build B of the application has different text in the field beside the check box. Functional Tester is looking for an object that matches the recognition properties recorded in Build A. We'll show how to fix this problem later in the tutorial.

6. When the log opens after playback, look at the messages. You should see two failures and one warning in the log. (Keep the log open in preparation for lesson 7.)

   The properties verification point (OrderForTrentCulpito_standa) and the image verification point (Album_image) 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? That difference did not cause the script to fail, however. The same objects are present but in a different location on the two applications. This did not cause a failure because Functional Tester 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.

Lesson 7: Use the Comparator to update a verification point

You can use the Verification Point Comparator to compare verification point data after you play back a script. 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.

At the end of lesson 6, you left the log open. If you closed the log, reopen it by double-clicking on the log name in the Projects view.

1. In the log, click the **View Results** link at the end of the failed image verification point entry. The event heading is ″Verification Point (Album_image).″

   The Functional Tester Verification Point Comparator displays your verification point data. Notice that the Comparator banner includes the name of your verification point.

Problems with the comparator?: If the comparator does not open or you get an error message, 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 Tester Help.

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.

When you created the verification point on ClassicsA, the captured album image is based on the object `javax.swing.JLabel`. When you played back the script on ClassicsB, since the height and the width of the object `javax.swing.JLabel` is different, the image verification point failed. So you must update the baseline file to change the object to match ClassicsB.

2. Click the **Load Baseline to Edit** button ( ) on the Comparator toolbar.
3. Click the **Replace Baseline with actual value** button ( ) on the Comparator toolbar. The actual image is loaded as the baseline image.
4. Close the Comparator.
5. In the log, click the **View Results** link at the end of the failed properties verification point entry. The event heading is "Verification Point (OrderforTrentCulpito_standard)."
6. 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 window. So you must update the baseline file to change the text to match ClassicsB.

   You can only edit the baseline file.
7. Click the **Load Baseline to Edit** button ( ) on the Comparator toolbar. Notice that the left **Value** column displays the **Baseline Value** now.
8. 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.
9. 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 Tester Help.
10. Close the Comparator.

    Now we will play back the script again to confirm the verification point passes, given the updated baseline value for the failure.
11. Close the log.
12. Click the **Run Functional Test Script** button on the Functional Tester toolbar.
13. Select the **Classics** log and then click **Finish**.
14. Click **Yes** if prompted to overwrite the log.

    Functional Tester pauses on the Member Logon window because you did not fix that recognition problem yet. At the end of playback, Functional Tester 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.
15. Leave the log open.

Lesson 8: Update the object map

In this lesson, 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 lesson 7, you left the log open. If it is not open, open it by double-clicking the log in the Projects view. One 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. Note the number and close the log to return to Functional Tester.
3. Click anywhere in the script window, and then click **Navigate → Go to Line**.
4. Type the line number from the log failure message, and then click **OK**.

   The cursor moves to the left margin of that line number.

   Note: You can also find the line number by looking at the indicator in the bottom of the Functional Tester window. For example: ″43:9″ refers to position 9 on line 43.
   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.
5. 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.

View the object recognition properties in the object map

1. Double-click the **rememberPassword** object to open it in the object map.
2. 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.
3. 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**.)
4. 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 **text** property. This is the ″old″ object. However, when you played back the script on ClassicsB, the text for that object changed, so Functional Tester 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.

Add the new object to the map

To add the new object to the map, open ClassicsB and then open the Member Logon window.

1. Click **Applications → Run** in the object map menu.
2. Select **ClassicsJavaB**. (Be sure to pick B).
3. Click **OK**.
4. In ClassicsCD, select any CD and then click **Place Order**.

The Member Logon window opens.

5. 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.
6. Clear the **After selecting an object advance to next page** check box if it is selected.
7. Use the Object Finder tool to select the **Remember the Password** check box in the Member Logon window.

   After you select the check box, you'll see that the **text** property is now `Remember The Password`. Stretch the borders of the object map, if necessary, to see the properties.
8. On the Select an Object page, click **Next**.
9. Don't change anything on the Select Object Options page, and then click **Finish**.

   The new check box object is now shown in the object map.
10. Click another object and notice that the new item is listed in blue and the word "New" is displayed 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.

Unify the objects

1. To unify the objects, click the old object (the original check box labeled **CheckBox: checkRemember**) and then 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. Then, release the mouse button.

   The Unify Test Objects wizard opens.
2. 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 Tester puts the new properties at the top if they are the preferred properties. However, some old administrative properties might be preferred. For example, Functional Tester 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.
3. Click **Next**.

   All scripts that are affected by this change in the object map are listed. Only one script, Classics, is affected.
4. Click **Finish**.
5. In the object map, click the **File → Save** menu on the object map toolbar to save the changes you made and then close the object map.

Play back the script again

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

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

Another way to update recognition properties: 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 Tester 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.

Lesson 9: Change the Recognition Preferences

In the previous lesson, 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 Tester 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 Tester 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. Select the **Use Default** check box beside this field.
5. Then type `5000` in the field and then 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.
8. Restore the default value for the recognition score:
   a. Click **Window → Preferences**.
   b. Click **Functional Test → Playback → ScriptAssure**.
   c. Click the **Advanced** button.
   d. Select the **Use Default** check box beside the **Warn if accepted score . . .** field.

      This will change the 5000 back to 10000.
   e. Click **OK**
   f. Play back the script again.

      Now the warning is gone and everything passes.
   g. Close the log.

This lesson 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 Tester Help.

Lesson 10: Use regular expressions

The last thing you will do using the object map is convert a property value to a regular expression. In this case, the regular expression provides more flexibility in the object recognition.

We just saw how the script passes completely on ClassicsB. That was the goal because the changes made to the application in ClassicsB are correct. So the script is now in the state you 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.

Open the object map and unify the objects

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 Tester 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 **rememberPassword** test object.
3. Close the log and then open the object map from the password check box object as you did in Lesson 8, by double-clicking **rememberPassword** in the Script Explorer.
4. In the object map, open the application by clicking **Applications → Run**. Select **ClassicsJavaA** and then click **OK**.
5. Pick any CD and click **Place Order** in ClassicsCD to open the Member Logon window.
6. Add the new object to the map by clicking **Test Object → Insert Object(s)**.
7. Use the Object Finder to select the password check box in the Member Logon window in the application.
8. Click **Next**, and then click **Finish**.
9. In the top pane of the object map, drag the old check box object to the new check box object to unify the objects.
10. 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`.

Convert a property value to a regular expression

1. In the top pane, right-click the `checkRemember` value and then click **Convert Value to Regular Expression**.

    Functional Tester designates the value as a regular expression by the ″xy″ icon in front of the value text.
2. Double-click the **name** value again so that you can edit the field.
3. Delete the word `check` and then edit the remainder to read: `[rR]ememberPassword`.
4. Click outside the 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 **accessibleContext.accessibleName** property is ″Remember Password″.
5. Right-click the Remember Password value and then select **Convert Value to Regular Expression** to convert it.

6. Double-click the value and edit it to read: `Remember.*Password`. You are removing the space and adding the period (.) and asterisk (*) characters.
7. 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.
8. Click **Next**, and then click **Finish**.
9. Click **File** → **Save** in the object map to save the changes, and then close the object map.
10. Close ClassicsCD.
11. Play back the script again on ClassicsA. In this case, the verification point is expected to fail because the text `Orders for Trent Culpito` was never changed to a regular expression. The object recognition warning on ClassicsA is no longer in the log.
12. Close the log.
13. Change the startApp command to play back ClassicsB, and then 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 Tester Help.

Summary: Create functional tests

This Functional Tester tutorial has shown you how to set up Functional Tester for testing, recording and playing back scripts, creating verification points and using the Verification Point Comparator to update object properties or data, and several ways to use the object map to your advantage.

Lessons learned

By completing this tutorial, you learned how to:

- Create a Functional Tester 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 Tester 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

Additional resources

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

- Functional Tester Help
- Functional Tester API Reference
- Functional Tester Welcome Page

Related information

ibm.com

eclipse.org