HP QuickTest Professional

Software Version: 10.00

User Guide
Volume 2

Manufacturing Part Number: T6513-90040
Document Release Date: January 2009
Software Release Date: January 2009

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Third-Party Web Sites

HP provides links to external third-party Web sites to help you find supplemental information. Site content and availability may change without notice. HP makes no representations or warranties whatsoever as to site content or availability.

Copyright Notices

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

Intel®, Pentium®, and Intel® Xeon™ are trademarks of Intel Corporation in the U.S. and other countries.

Java™ is a US trademark of Sun Microsystems, Inc.

Microsoft®, Windows®, Windows NT®, and Windows® XP are U.S registered trademarks of Microsoft Corporation.

Oracle® is a registered US trademark of Oracle Corporation, Redwood City, California.

Unix® is a registered trademark of The Open Group.

SlickEdit® is a registered trademark of SlickEdit Inc.

Documentation Updates

This guide's title page contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP Software Support web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software Support Online provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the HP Software Support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Table of Contents

This Table of Contents lists all of the chapters in both volumes of the *HP QuickTest Professional User Guide*.

PART IV: ENHANCING TESTS (VOL. 1)

PART VII: MAINTAINING AND DEBUGGING TESTS (VOL. 2)

PART VIII: WORKING WITH THE QUICKTEST IDE (VOL. 2)

PART X: WORKING WITH ADVANCED TESTING FEATURES (VOL. 2)

PART XI: WORKING WITH QUALITY CENTER (VOL. 2)

PART XII: WORKING WITH OTHER HP PRODUCTS (VOL. 2)

Welcome to This Guide

Welcome to the *HP QuickTest Professional User Guide*. This guide describes how to use QuickTest to test your applications. It provides step-by-step instructions to help you create, debug, and run tests, and report defects detected during the testing process.

This chapter includes:

How This Guide Is Organized

The QuickTest Professional User Guide is divided into two volumes in the printed version. In the PDF and context-sensitive Help versions of this guide, which are included with the QuickTest Professional installation, the information from both volumes is combined into a single file.

This guide contains the following parts:

Volume 1

Part I Introducing QuickTest Professional

Provides an overview of QuickTest and the main stages of the testing process.

Part II Working with Test Objects

Introduces the test object model and describes how QuickTest identifies objects in your application. It describes how to work with objects, configure object identification, and create Smart Identification definitions. It also describes how to manage, merge, and compare object repositories.

Part III Designing Tests

Describes how to plan and create tests, and how to work with actions.

Part IV Enhancing Tests

Describes how to insert checkpoints, parameters, and output values, and use regular expressions.

Volume 2

Part V Defining Functions and Other Programming Tasks

Describes how to enhance your test using the Expert View, how to customize the Expert View and function library windows, and how to work with user-defined functions and function libraries in QuickTest.

Part VI Running and Analyzing Tests

Describes how to run tests and analyze the results.

Part VII Maintaining and Debugging Tests

Describes how to control run sessions to identify and isolate bugs in test scripts and function libraries.

Part VIII Working with the QuickTest IDE

Describes how to modify the QuickTest layout, how to manage testing resources, and how to work with process guidance.

Part IX Configuring QuickTest Settings

Describes how to modify global and local QuickTest testing options, and how to set testing options during a run session.

Part X Working with Advanced Testing Features

Describes how to work with virtual objects and recovery scenarios. It also describes several programming techniques to create more powerful scripts, and describes how to automate QuickTest operations.

Part XI Working with Quality Center

Describes how to integrate and work with HP Quality Center, which provides an intuitive and efficient method for running tests, collecting and analyzing test results, tracking defects, and managing test versions.

Part XII Working with Other HP Products

Describes how you can run tests and call functions in compiled modules from WinRunner, the HP enterprise functional testing tool for Microsoft Windows applications. This section also describes how to use QuickTest with Business Process Testing, and how QuickTest interacts with Quality Center, the HP centralized quality solution. This section also describes considerations for designing QuickTest tests for use with HP performance testing and application management products.

Part XIII Appendixes

Provides information on frequently asked questions, supported checkpoints and output values, creating customized process guidance packages, and customizing the algorithm used to compare bitmaps in bitmap checkpoints.

Who Should Read This Guide

This guide is intended for QuickTest Professional users at all levels. Readers should already have some understanding of functional testing concepts and processes, and know which aspects of their application they want to test.

QuickTest Professional Online Documentation

QuickTest Professional includes the following online documentation:

Readme provides the latest news and information about QuickTest. Select **Start** > **Programs** > **QuickTest Professional** > **Readme**.

HP QuickTest Professional Installation Guide explains how to install and set up QuickTest. Select **Help** > **Printer-Friendly Documentation** > **HP QuickTest Professional Installation Guide**.

HP QuickTest Professional Tutorial teaches you basic QuickTest skills and shows you how to design tests for your applications. Select **Help** > **QuickTest Professional Tutorial**.

Product Feature Movies provide an overview and step-by-step instructions describing how to use selected QuickTest features. Select **Help** > **Product Feature Movies**.

Printer-Friendly Documentation displays the complete documentation set in Adobe portable document format (PDF). Online books can be viewed and printed using Adobe Reader, which can be downloaded from the Adobe Web site (http://www.adobe.com). Select **Help** > **Printer-Friendly Documentation**.

QuickTest Professional Help includes:

- **What's New in QuickTest Professional** describes the newest features, enhancements, and supported environments in the latest version of QuickTest.
- **HP QuickTest Professional User Guide** describes how to use QuickTest to test your application.
- **HP QuickTest Professional for Business Process Testing User Guide** provides step-by-step instructions for using QuickTest to create and manage assets for use with Business Process Testing.
- **HP QuickTest Professional Add-ins Guide** describes how to work with supported environments using QuickTest add-ins, and provides environment-specific information for each add-in.
- **HP QuickTest Professional Object Model Reference** describes QuickTest test objects, lists the methods and properties associated with each object, and provides syntax information and examples for each method and property.

- **HP QuickTest Professional Advanced References** contains documentation for the following QuickTest COM and XML references:
  - **HP QuickTest Professional Automation Object Model** provides syntax, descriptive information, and examples for the automation objects, methods, and properties. It also contains a detailed overview to help you get started writing QuickTest automation scripts. The automation object model assists you in automating test management, by providing objects, methods and properties that enable you to control virtually every QuickTest feature and capability.
  - **HP QuickTest Professional Test Results Schema** documents the test results XML schema, which provides the information you need to customize your test results.
  - **HP QuickTest Professional Test Object Schema** documents the test object XML schema, which provides the information you need to extend test object support in different environments.
  - **HP QuickTest Professional Object Repository Schema** documents the object repository XML schema, which provides the information you need to edit an object repository file that was exported to XML.
  - **HP QuickTest Professional Object Repository Automation** documents the Object Repository automation object model, which provides the information you need to manipulate QuickTest object repositories and their contents from outside of QuickTest.
- **VBScript Reference** contains Microsoft VBScript documentation, including VBScript, Script Runtime, and Windows Script Host.

To access the QuickTest Professional Help, select **Help** > **QuickTest Professional Help**. You can also access the QuickTest Professional Help by clicking in selected QuickTest windows and dialog boxes and pressing F1. Additionally, you can view a description, syntax, and examples for a QuickTest test object, method, or property by placing the cursor on it and pressing F1.

Additional Online Resources

Mercury Tours sample Web site is the basis for many examples in this guide. The URL for this Web site is http://newtours.demoaut.com. Select **Start** > **Programs** > **QuickTest Professional** > **Sample Applications** > **Mercury Tours Web Site**.

The **HP Software Web site** provides you with the most up-to-date information on HP Software products. This includes new software releases, seminars and trade shows, customer support, and more. The URL for this Web site is www.hp.com/go/software.

The following additional online resources are available from the QuickTest Professional **Help** menu:

Troubleshooting & Knowledge Base accesses the Troubleshooting page on the HP Software Support Web site where you can search the Self-solve knowledge base. Choose **Help** > **Troubleshooting & Knowledge Base**. The URL for this Web site is http://h20230.www2.hp.com/troubleshooting.jsp.

HP Software Support accesses the HP Software Support Web site. This site enables you to browse the Self-solve knowledge base. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. Choose **Help** > **HP Software Support**. The URL for this Web site is www.hp.com/go/hpsoftwaresupport.

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport user ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Part V

Defining Functions and Other Programming Tasks

29

Working in the Expert View and Function Library Windows

In QuickTest, tests are composed of statements coded in the Microsoft VBScript programming language. The Expert View provides an alternative to the Keyword View for testers who are familiar with VBScript. You can also create function libraries in QuickTest using VBScript.

This chapter explains how to work in the Expert View, provides a brief introduction to VBScript, and shows you how to enhance your tests and function libraries using a few simple programming techniques.

This chapter includes:

About Working in the Expert View and Function Library Windows

In the Expert View, you can view an action in VBScript. If you are familiar with VBScript, you can add and update statements and enhance your tests and function libraries with programming. This enables you to increase a test's power and flexibility. You can also create and work with function libraries using the Function Library window.

To learn about working with VBScript, you can view the VBScript documentation directly from the QuickTest **Help** menu (**Help > QuickTest Professional Help > VBScript Reference**).

You can add statements that perform operations on objects or retrieve information from your application. For example, you can add a step that checks that an object exists, or you can retrieve the return value of an operation.

You can add steps to your test or function library either manually or using the Step Generator. For more information on using the Step Generator, see "Inserting Steps Using the Step Generator" on page 777.

You can print the test displayed in the Expert View or a function library at any time. You can also include additional information in the printout. For more information on printing from the Expert View, see "Printing a Test" on page 332. For more information on printing a function library, see "Printing a Function Library" on page 917.

Understanding and Using the Expert View

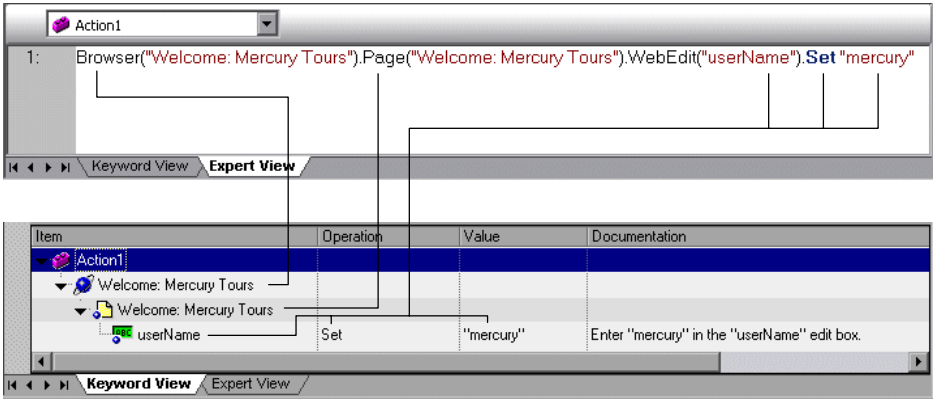
If you prefer to work with VBScript statements, you can choose to work with your tests in the Expert View, as an alternative to using the Keyword View. You can move between the two views as you wish, by selecting the Expert View or Keyword View tab at the bottom of the Test pane in the QuickTest window.

Working in the Expert View

The Expert View displays the same steps and objects as the Keyword View, but in a different format:

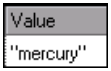
- In the Keyword View, QuickTest displays information about each step and shows the object hierarchy in an icon-based table. For more information, see Chapter 14, "Working with the Keyword View."
- In the Expert View, QuickTest displays each step as a VBScript line or statement. In object-based steps, the VBScript statement defines the object hierarchy.

  The following diagram shows how the same object hierarchy is displayed in the Expert View and in the Keyword View:



Each line of VBScript in the Expert View represents a step in the test. The example above represents a step in a test in which the user inserts the name mercury into an edit box. The hierarchy of the step enables you to see the name of the site, the name of the page, the type and name of the object in the page, and the name of the operation performed on the object.

The table below explains how the different parts of the same step are represented in the Keyword View and the Expert View:

Keyword View	Expert View	Explanation
	Browser ("Welcome: Mercury Tours")	The name of the browser test object is Welcome: Mercury Tours.
	Page ("Welcome: Mercury Tours")	The name of the current page is Welcome: Mercury Tours.
	WebEdit ("userName")	The object type is WebEdit; the name of the edit box on which the operation is performed is userName.
	Set	The method performed on the edit box is **Set**.
	"mercury"	The value inserted into the **username** edit box is mercury.

In the Expert View, an object's description is displayed in parentheses following the object type. For all objects stored in the object repository, the object name is a sufficient object description. In the following example, the object type is **Browser**, and the object name is **Welcome: Mercury Tours**:

```
Browser ("Welcome: Mercury Tours")
```

Tip: Test object and operation names are not case sensitive.

The objects in the object hierarchy are separated by a dot. In the following example, **Browser** and **Page** are two separate objects in the same hierarchy:

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours")
```

The operation (method) performed on the object is always displayed at the end of the statement, followed by any values associated with the operation. In the following example, the word **mercury** is inserted in the **userName** edit box using the **Set** method:

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    WebEdit("userName").Set "mercury"
```

QuickTest relates to your application in terms of the objects in it. The steps you add to your test correspond to the operations performed on the objects in your application.

The objects in QuickTest are divided by environment. By default, QuickTest supports objects from the standard Windows environments. You can work with additional environments by loading the relevant QuickTest add-ins in the Add-in Manager when you open QuickTest.

Most objects have corresponding operations. For example, the **Back** method is associated with the **Browser** object.

For a complete list of objects and their associated operations and properties, select **Help > QuickTest Professional Help**, and open the **QuickTest Object Model** from the Contents tab.

For more information on adding steps that perform operations, see "Generating Statements in the Expert View or in a Function Library" on page 833.

For more information on using VBScript, see "Understanding Basic VBScript Syntax" on page 853.

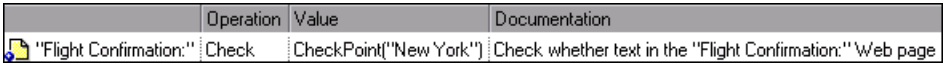
Understanding Checkpoint and Output Statements

In QuickTest, you can create checkpoints and output values on pages, text strings, tables, and other objects. When you create a checkpoint or output value in the Keyword View, QuickTest creates a corresponding line in VBScript in the Expert View. It uses the **Check** method to perform the checkpoint, and the **Output** method to perform the output value step.

For example, in the following statement QuickTest performs a check on the words New York:

```
Browser("Mercury Tours").Page("Flight Confirmation").Check
    Checkpoint("New York")
```

The corresponding step in the Keyword View is displayed as follows:



	Operation	Value	Documentation
"Flight Confirmation:"	Check	CheckPoint("New York")	Check whether text in the "Flight Confirmation:" Web page

---

Notes:

- The details about a checkpoint are set in the relevant Checkpoint Properties dialog box and are stored with the object it checks. The details about an output value step are set in the relevant Output Value Properties dialog box and are stored with the object whose values it outputs. The statement displayed in the Expert View is a reference to the stored information. Therefore, you cannot insert a checkpoint or output value statement in the Expert View manually and you cannot copy a **Checkpoint** or **Output** statement from the Expert View to another test.
- For more information on inserting and modifying checkpoints, see Chapter 17, "Understanding Checkpoints." For more information on inserting and modifying output values, see Chapter 25, "Outputting Values."

Understanding Parameter Indications

You can use QuickTest to enhance your tests by parameterizing values. A **parameter** is a variable that is assigned a value from an external data source or generator.

When you create a parameter in the Keyword View, QuickTest creates a corresponding line in VBScript in the Expert View.

For example, if you define the value of a method argument as a Data Table parameter, QuickTest retrieves the value from the Data Table using the following syntax:

Object_Hierarchy.Method **DataTable (***parameterID***,** *sheetID***)**

Item	Description
Object_Hierarchy	The hierarchical definition of the test object, consisting of one or more objects separated by a dot.
Method	The name of the method that QuickTest executes on the parameterized object.
DataTable	The reserved object representing the Data Table.
parameterID	The name of the column in the Data Table from which to take the value.
sheetID	The name of the sheet in which the value is stored. If the parameter is a global parameter, dtGlobalSheet is the sheet ID.

For example, suppose you are creating a test for the Mercury Tours site, and you select San Francisco as your destination. The following statement would be inserted into your test in the Expert View:

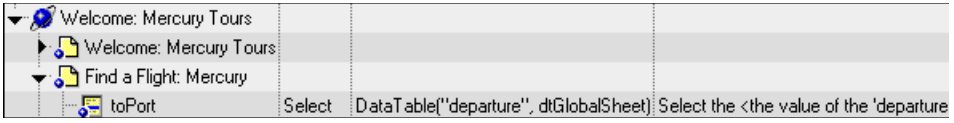
```
Browser("Welcome: Mercury").Page("Find a Flight:").WebList("toPort").
    Select "San Francisco"
```

Now suppose you parameterize the destination value, and you create a **Destination** column in the Data Table. The previous statement would be modified to the following:

```
Browser("Welcome: Mercury").Page("Find a Flight:").WebList("toPort").
    Select DataTable("Destination",dtGlobalSheet)
```

In this example, **Select** is the method name, DataTable is the object that represents the Data Table, Destination is the name of the column in the Data Table, and dtGlobalSheet indicates the Global sheet in the Data Table.

In the Keyword View, this step is displayed as follows:



For more information on using and defining parameter values, see Chapter 24, “Parameterizing Values.”

Generating Statements in the Expert View or in a Function Library

You can generate statements in the following ways:

- You can use the Step Generator to add steps that use methods and functions. For more information, see "Inserting Steps Using the Step Generator" on page 777.
- You can manually insert VBScript statements that perform operations. QuickTest includes features that help you adhere to the correct syntax and select the relevant items for your statements.
  - **Statement completion (IntelliSense).** This option, when enabled, helps you select the variable, test object, operation, property, or collection for your statement and view the relevant syntax as you type in the Expert View or a function library. For more information, see "Using Statement Completion (IntelliSense)" on page 833.
  - **Auto-expand VBScript syntax.** When this option is enabled, QuickTest automatically adds the relevant syntax or blocks to your script, when you start to type a VBScript keyword in the Expert View or in a function library. For more information, see "Automatically Completing VBScript Syntax" on page 842.

Using Statement Completion (IntelliSense)

When you type in the Expert View or a function library, IntelliSense (the statement completion feature included with QuickTest) enables you to select the variable, test object, operation, property, or collection for your statement from a drop-down list and view the relevant syntax.

The **Statement Completion** option is enabled by default. You can disable or enable this option in the Editor Options dialog box. For more information, see Chapter 30, "Customizing the Expert View and Function Library Windows."

Tips:

- In some cases, QuickTest needs to retrieve IntelliSense information from an actual object. In such cases, you may experience a delay while typing in the Expert View or a function library. To avoid this delay, you can disable the statement completion option.
- Although IntelliSense in function library documents is supported to help generate test object statements, as described below, it is generally not recommended to include a full object hierarchy statement in a function. It is preferable to make your functions generic so that they can be used with different objects.
- QuickTest might not display IntelliSense information if the statement is typed incorrectly and contains syntax errors or other VBScript errors.
- If you resize the frame in which the IntelliSense drop-down list is displayed, QuickTest subsequently uses the new size when it displays IntelliSense drop-down lists.
- To close the IntelliSense drop-down list without selecting from it, press ESC.

When the **Statement Completion** option is enabled it provides the following types of information:

- **Available test objects.** If you type a test object class followed by an open parenthesis **(** , QuickTest displays a list of all test objects of this class in the object repository. If there is only one object of this class in the object repository, QuickTest automatically enters its name in quotes after the open parenthesis. For example, if you type `Page(`, QuickTest displays a list of all the **Page** test objects in the object repository.
- **Available operations and properties.** If you type a period after an object or test object in a statement, QuickTest displays a list of the operations and properties that you can add after the object you typed.

  As you type the name of an operation or a property, QuickTest highlights the first operation or property (alphabetically) that matches the text you typed. Pressing ENTER or SPACE enters the highlighted word in the step.

Tip: If you type the name of an operation or property when the list of available operations and properties is not displayed, pressing CTRL+SPACE automatically completes the word if there is only one option, or displays the list and highlights the first operation or property (alphabetically) that matches the text you typed. Pressing ENTER or SPACE enters the highlighted word in the step.

QuickTest provides this type of IntelliSense information, when available, for test objects, reserved objects, objects you create in your test or function, variables to which objects or test objects are assigned, and properties or operations which return objects.

For example:

- If you type a period after a test object in a statement, QuickTest displays a list of the relevant test objects, operations, properties, collections, and registered functions that you can add after the object you typed.
- If you type a period after an object that you created in your script (using the **CreateObject** method, for example), QuickTest displays the operations and properties that you can use for that object.
- If you use the **Object** property in your statement and the object data is currently available in the Active screen or the open application, QuickTest displays the native operations and properties of the object. For more information on the **Object** property, see "Accessing Native Properties and Operations" on page 887.
- If you type a period within a **With** statement, QuickTest displays a list of the operations and properties available for the relevant object.

Note: If you type a **With** statement (as opposed to using a menu command to create it), you must use the **Edit** > **Advanced** > **Apply "With" to Script** command (or press CTRL+W) to enable IntelliSense within the **With** statement.

➤ If you assign an object to a variable, and then type the name of the variable followed by a period, QuickTest displays a list of the operations and properties available for the object.

In some cases, the value of a variable cannot be determined while editing the test (for example, if the value is set by a conditional assignment or returned by another function). In this case, QuickTest provides IntelliSense information according to the most recent line of code in which the value of the variable could be evaluated, if any.

The following examples illustrate this:

Example 1:

```
Line 1: Set x = CreateObject("Excel.Application")
Line 2: z = GetValueFromUser()
Line 3: If z = 2 Then
Line 4:    Set x = CreateObject("Word.Application")
Line 5: End If
Line 6: x.
```

While editing this test, QuickTest cannot determine which object will actually be assigned to x in line 6. However, because the value of x can be evaluated independently in line 4, QuickTest displays the IntelliSense information relevant to the object "Word.Application" for the variable x in line 6.

Example 2:

```
Line 1: Set x = CreateObject("Excel.Application")
Line 2: Set x = MyGetObject()
Line 3: x.
```

While editing this test, QuickTest cannot determine the type of object that the **MyGetObject** function returns (line 2). Therefore, in line 3 in the example above, QuickTest displays the IntelliSense information relevant to the object "Excel.Application", because line 1 is the most recent line of code in which the value of x could be evaluated. However, if line 2 were not preceded by a line in which the value could be evaluated, QuickTest would not display any IntelliSense information for x in line 3.

➤ **Operation or property syntax.** If you type a space after the name of an operation or property, QuickTest displays the syntax for it, including its mandatory and optional arguments. When you add a step that uses an operation or property, you must define a value for each mandatory argument associated with the operation or property.

When you type a comma after an argument value (other than the last one in the step), QuickTest displays the operation syntax again, bolding the next argument for which you need to enter a value.

You can also place the cursor on any operation or function that contains arguments and press CTRL+SHIFT+SPACE or select **Edit** > **Advanced** > **Argument Info** to display the statement completion (argument syntax) tooltip for that item.

➤ **Possible argument values.** For certain operations, when you type the space or comma before an argument that has a predefined list of values, QuickTest displays the list of possible values. In the Expert View, when working with Java or ActiveX objects, QuickTest dynamically retrieves the list of possible values for certain arguments from the object in the application. For QuickTest to retrieve the possible values, the application must be open and the relevant object must be visible. For example, QuickTest can retrieve the list of items in a specific Java list object, and display them as the possible values for the **Item** argument of the **Select** method.

Note: When you edit a test during a recording session, QuickTest does not retrieve the possible argument values from the application.

➤ **Available constants and local variables.** If you begin to type a constant or a local variable name, QuickTest displays a list of constants and local variables (relevant to the current programming scope) that begin with the letters you typed. If there is only one matching constant or variable defined, QuickTest automatically enters its name in the step.

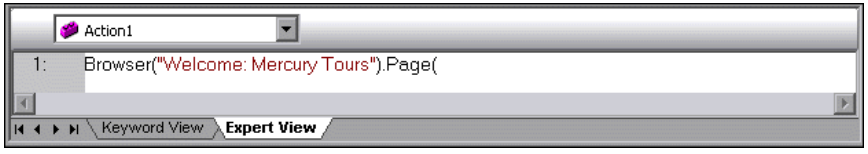
Tip: If you press CTRL+SPACE, QuickTest displays a list of the relevant test objects, operations, properties, collections, VBScript functions, user-defined functions, VBScript constants, and utility objects that you can add. This list is displayed even if you typed an object that has not yet been added to the object repository. If the test contains a function, or is associated with a function library, the functions are also displayed in the list.

To generate a statement using statement completion in the Expert View or a function library:

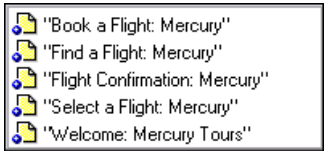
1 Confirm that the **Statement completion** option is selected (**Tools** > **View Options** > **General** tab).

2 Perform one of the following:

- If you are working in a function library, skip to step 4
- If you are working in the Expert View, type an object followed by an open parenthesis **(**



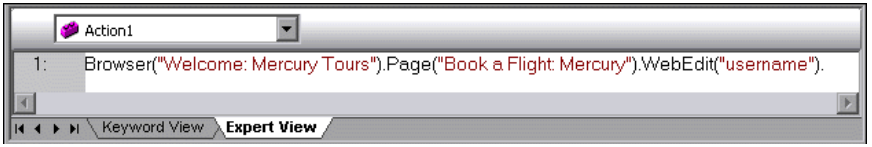
If there is only one object of this type in the object repository, QuickTest automatically enters its name in quotes after the open parenthesis. If more than one object of this type exists in the object repository, QuickTest displays them in a list.



3 Double-click an object in the list or use the arrow keys to choose an object and press ENTER. QuickTest inserts the object into the statement.

4 Perform one of the following:

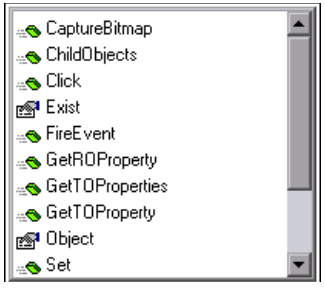
➤ If you are working in the Expert View, type a period (.) after the object on which you want to perform the operation.



➤ If you are working in a function library, type the full hierarchy of an object, for example:

```
Browser("Welcome: Mercury Tours").Page("Book a Flight:
   Mercury).WebEdit("username")
```

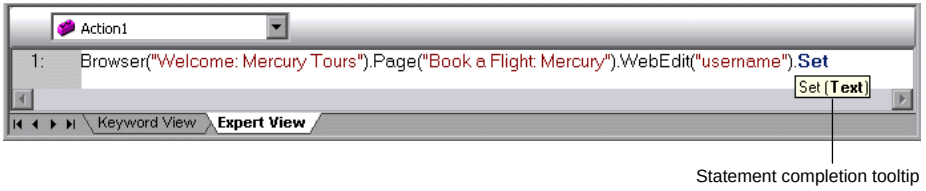
5 Type a period (.) after the object description, for example ("username"). QuickTest displays a list of the available operations and properties for the object.



Tip:

- As you type the name of an operation or property, QuickTest highlights the first operation or property (alphabetically) that matches the text you typed. Pressing ENTER or SPACE inserts the highlighted word in the step.
- If you type the name of an operation or property when the list of available operations and properties is not displayed, you can press CTRL+SPACE or select **Edit** > **Advanced** > **Complete Word**. If only one operation or property matches the text you typed, QuickTest automatically completes the operation or property name. Otherwise, QuickTest displays the list and highlights the first operation or property (alphabetically) that matches the text you typed. Pressing ENTER or SPACE inserts the highlighted word in the step.

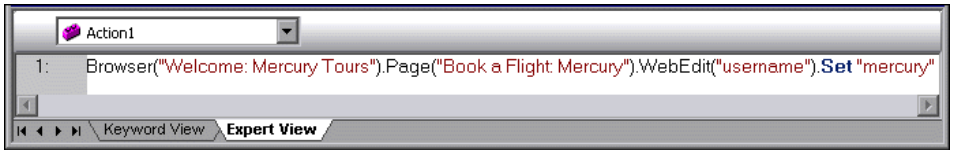
6 Double-click an operation or property in the list or use the arrow keys to choose an operation or property and press ENTER. QuickTest inserts the operation or property into the statement. If the operation or property contains arguments, QuickTest displays the syntax of the operation or property in a tooltip, as shown in this example from the Expert View.



In the above example, the **Set** method has one argument, called **Text**. The argument name represents the text to insert in the box.

Tip: You can also place the cursor on any operation or function that contains arguments and press CTRL+SHIFT+SPACE or select **Edit** > **Advanced** > **Argument Info** to display the statement completion (argument syntax) tooltip for that item.

7 Enter the operation arguments after the operation according to the displayed syntax.



Note: After you have added a step in the Expert View, you can view the new step in the Keyword View. If the statement that you added in the Expert View contains syntax errors, QuickTest displays the errors in the Information pane when you select the Keyword View. For more information, see "Handling VBScript Syntax Errors" on page 860.

For more details and examples of any QuickTest operation, see the *HP QuickTest Professional Object Model Reference*.

For more information on VBScript syntax, see "Understanding Basic VBScript Syntax" on page 853.

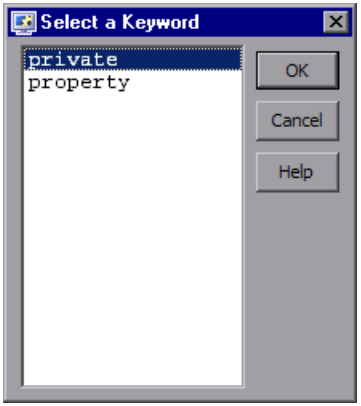
Automatically Completing VBScript Syntax

When the **Auto-expand VBScript syntax** option is enabled and you start to type a VBScript keyword in the Expert View or a function library, QuickTest automatically recognizes the first two characters of the keyword and adds the relevant VBScript syntax or blocks to the script. For example, if you enter the letters if and then enter a space at the beginning of an empty line, QuickTest automatically enters:

```
If Then
End If
```

The **Auto-expand VBScript syntax** option is enabled by default. You can disable or enable this option in the Editor Options dialog box. For more information, see "Customizing Editor Behavior" on page 897.

If you enter two characters that are the initial characters of multiple keywords, the Select a Keyword dialog box is displayed and you can select the keyword you want. For example, if you enter the letters pr and then enter a space, the Select a Keyword dialog box opens containing the keywords private and property.



You can then select a keyword from the list and click **OK**. QuickTest automatically enters the relevant VBScript syntax or block in the script.

For more information on VBScript syntax, see "Understanding Basic VBScript Syntax" on page 853.

Navigating in the Expert View and Function Libraries

You can use the Go To dialog box or bookmarks to jump to a specific line in the Expert View or a function library. You can also find specific text strings in the Expert View or a function library, and, if desired, replace them with different strings. These options make it easier to navigate through sections of a long action or function.

Note: When working with tests, the Expert View displays only one action. The navigation features described in this section are available only for the currently selected action and not for the entire test.

Using the Go To Dialog Box

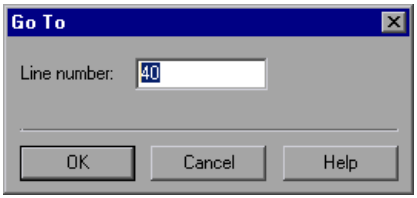
You can use the Go To dialog box to navigate to a specific line in an action or in a function library.

Tip: By default, line numbers are displayed in the Expert View and in function libraries. If they are not displayed, you can select the **Show line numbers** option in the **Tools > View Options > General** tab. For more information on the Editor options, see Chapter 30, "Customizing the Expert View and Function Library Windows."

To navigate to a line in the Expert View or a function library using the Go To dialog box:

1 Click the **Expert View** tab or activate a function library.

2 Select **Edit** > **Go To**. The Go To dialog box opens.



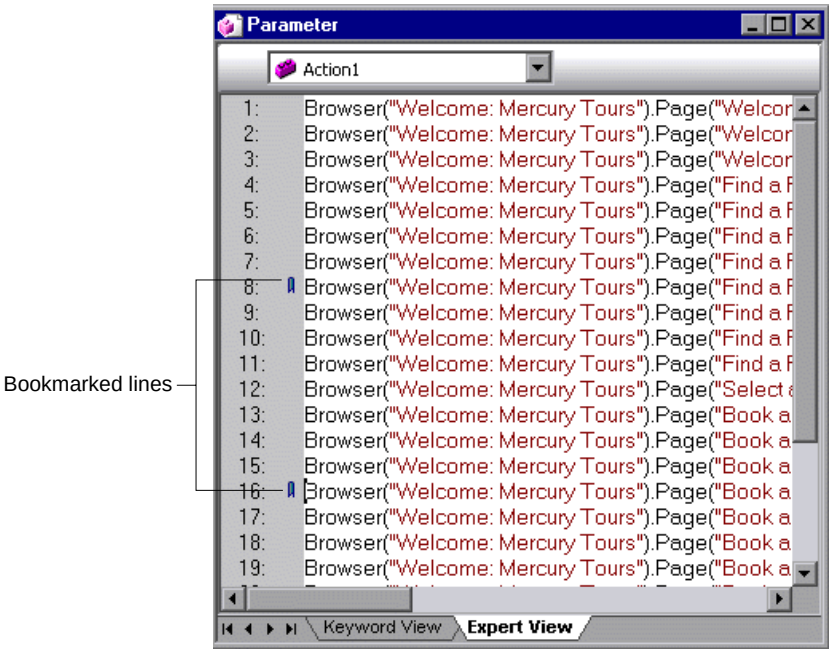
3 Enter the line to which you want to navigate in the **Line number** box and click **OK**. The cursor moves to the beginning of the line you specify.

Working with Bookmarks

You can use bookmarks to mark important sections in your action or function library so that you can navigate between the various parts more easily. In tests, bookmarks apply only within a specific action; they are not preserved when you navigate between actions and they are not saved with the test or function library.

When you assign a bookmark, an icon is added to the left of the selected line in the Expert View or function library. You can then use the **Go To** button in the Bookmarks dialog box to jump to the bookmarked rows.

Bookmarks look the same in tests and in function libraries. In the following example, two bookmarks have been added to an action in a test.

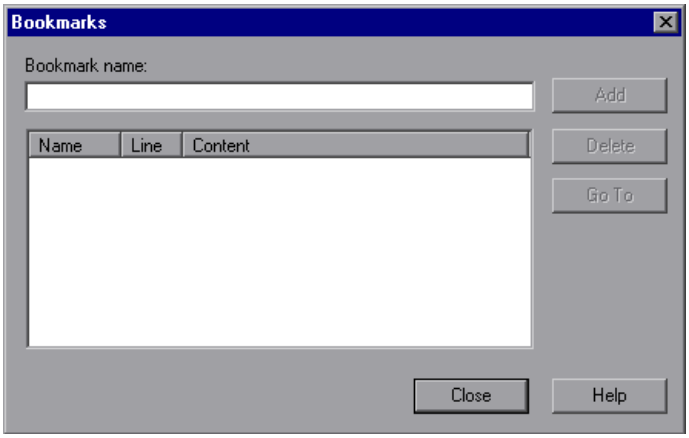


To set bookmarks:

1 Click the **Expert View** tab or activate a function library.

2 Click in the line to which you want to assign a bookmark.

3 Select **Edit** > **Bookmarks**. The Bookmarks dialog box opens.



4 In the **Bookmark name** field, enter a unique name for the bookmark and click **Add**. The bookmark is added to the Bookmarks dialog box, together with the line number at which it is located and the textual content of the line. In addition, a bookmark icon is added to the left of the selected line in the Expert View or function library.

5 To delete a bookmark, select it in the list and click **Delete**.

To navigate to a specific bookmark:

1 Click the **Expert View** tab or activate a function library.

2 Select **Edit** > **Bookmarks**. The Bookmarks dialog box opens.

3 Select a bookmark from the list and click the **Go To** button. QuickTest jumps to the appropriate line in the current action or function library.

Finding Text Strings

You can specify text strings to locate in the current action in the Expert View or in a function library. You can also search for strings in the Edit HTML Source and Edit HTML Tags dialog boxes of Page checkpoints, and in the "With" Generation Results dialog box. You can either search for literal text or use regular expressions for a more advanced search. You can also use other options to further fine-tune your search results.

For more information on the With Generation Results dialog box, see "Generating With Statements for Your Test" on page 806. For more information on Page checkpoints, see the section on Page checkpoints in the *HP QuickTest Professional Add-ins Guide*.

To find a text string:

1 In the Expert View or function library, perform one of the following:

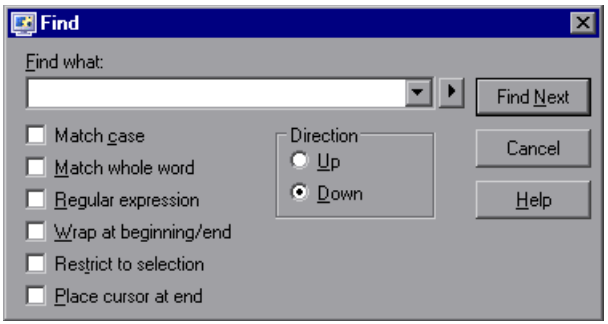
- Click the **Find** button.
- Select **Edit** > **Find**.

Tip: In the Expert View, you can also perform one of the following:

Select **Edit** > **Advanced** > **Apply "With" to Script**, and then press CTRL+F.

In the Page Checkpoint Properties dialog box, click **Edit HTML Source** or **Edit HTML Tags**, and then right-click and select **Find** in the displayed dialog box.

The Find dialog box opens.



2 In the **Find what** box, enter the text string you want to locate.

3 If you want to use regular expressions in the string you specify, click the arrow button (▶) and select a regular expression. When you select a regular expression from the list, it is automatically inserted in the **Find what** box at the cursor location. For more information, see "Using Regular Expressions in the Find and Replace Dialog Boxes" on page 852.

4 Select any of the following options to help fine-tune your search:

- **Match case.** Distinguishes between upper-case and lower-case characters in the search. When **Match case** is selected, QuickTest finds only those occurrences in which the capitalization matches the text you entered in the **Find what** box exactly.
- **Match whole word.** Searches for occurrences that are only whole words and not part of longer words.
- **Regular expression.** Treats the specified text string as a regular expression. This option is automatically selected when you select a regular expression from the list.
- **Wrap at beginning/end.** Continues the search from the beginning or end of the action, dialog box, or function library text when either the beginning or end is reached, depending on the selected search direction.
- **Restrict to selection.** Searches only within the selected part of the action, dialog box, or function library text.
- **Place cursor at end.** Places the cursor at the end of the highlighted occurrence when the search string is located.

5 Specify the direction in which you want to search, from the current cursor location in the action, dialog box, or function library: **Up** or **Down**

6 Click **Find Next** to highlight the next occurrence of the specified string in the current action, dialog box, or in the active function library.

Replacing Text Strings

You can specify text strings to locate in the current action in the Expert View or function library, and specify the text strings you want to use to replace them. You can also search and replace strings in the Edit HTML Source and Edit HTML Tags dialog boxes. You can either find and replace literal text or use regular expressions for a more advanced process. You can also use other options to further fine-tune your find and replace process.

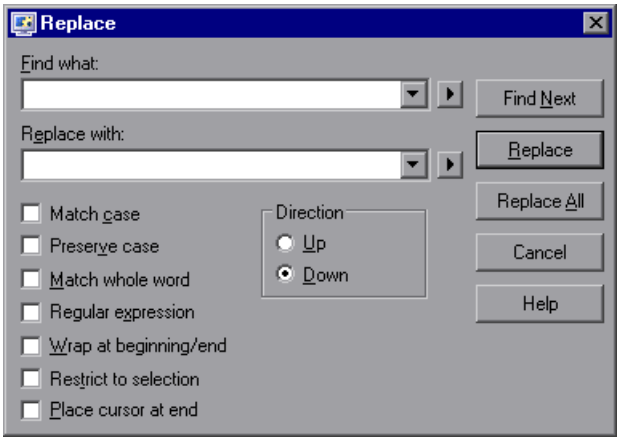
To replace a text string:

1 In the Expert View or function library, perform one of the following:

- Click the **Replace** button.
- Select **Edit** > **Replace**.

Tip: In the Page Checkpoint Properties dialog box, click **Edit HTML Source** or **Edit HTML Tags**, and then right-click and select **Replace** in the displayed dialog box.

The Replace dialog box opens.



2 In the **Find what** box, enter the text string you want to locate.

3 In the **Replace with** box, enter the text string you want to use to replace the found text.

4 If you want to use regular expressions in the **Find what** or **Replace with** string, click the arrow button (▶) and select a regular expression. When you select a regular expression from the list, it is automatically inserted in the **Find what** or **Replace with** box at the cursor location. For more information, see "Using Regular Expressions in the Find and Replace Dialog Boxes" on page 852.

5 Select any of the following options to help fine-tune your search:

- **Match case.** Distinguishes between upper-case and lower-case characters in the search. When **Match case** is selected, QuickTest finds only those occurrences in which the capitalization exactly matches the text you entered in the **Find what** box.
- **Preserve case.** Checks each occurrence of the **Find what** string for all lowercase, all uppercase, sentence caps or mixed case. The **Replace with** string is converted to the same case as the occurrence found, except when the occurrence found is mixed case. In this case, the **Replace with** string is used without modification.
- **Match whole word.** Searches for occurrences that are whole words only and not part of longer words.

- **Regular expression.** Treats the specified text string as a regular expression. This option is automatically selected when you select a regular expression from the list.
- **Wrap at beginning/end.** Continues the search from the beginning or end of the action, dialog box, or function library text when either the beginning or end is reached, depending on the selected search direction.
- **Restrict to selection.** Searches only within the selected part of the action, dialog box, or function library text.
- **Place cursor at end.** Places the cursor at the end of the highlighted occurrence when the search string is located.
- **Direction.** Specifies the search direction.
  - **Up.** Searches only from the current text up to the beginning of the action, dialog box, or function library text.
  - **Down.** Searches only from the current text down to the end of the action, dialog box, or function library text.

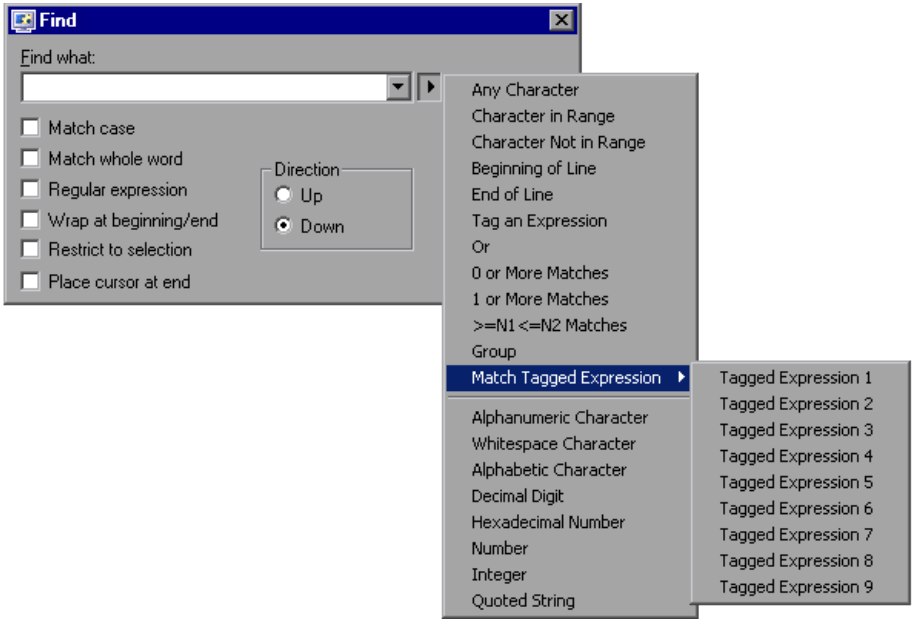
6 Click **Find Next** to highlight the next occurrence of the specified text string in the current action or dialog box, or in the active function library.

7 Click **Replace** to replace the highlighted text with the text in the **Replace with** box, or click **Replace All** to replace all occurrences specified in the **Find what** box with the text in the **Replace with** box in the current action or dialog box, or in the active function library.

Using Regular Expressions in the Find and Replace Dialog Boxes

You can use regular expressions in the **Find what** and **Replace with** strings to enhance your search. For a general understanding of regular expressions, see "Understanding and Using Regular Expressions" on page 762. Note that there are differences in the expressions supported by the Find and Replace dialog boxes and the expressions supported in other parts of QuickTest.

You display the regular expressions available for selection by clicking the arrow button ▸ in the Find or Replace dialog boxes.



You can select from a predefined list of regular expressions. You can also use tagged expressions. When you use regular expressions to search for a string, you may want the string to change depending on what was already found.

For example, you can search for **(save\:n)\1**, which will find any occurrence of **save** followed by any number, immediately followed by **save**, as well as the same number that was already found (meaning that it will find **save6save6** but not **save6save7**).

You can also use tagged expressions to insert parts of what is found into the replace string. For example, you can search for **save(\:n)** and replace it with **open\1**. This will find **save** followed by any number, and replace it with **open** and the number that was found.

Select **Tag an Expression** from the regular expressions list to insert parentheses "**()**" to indicate a tagged expression in the search string.

Select **Match Tagged Expression** and then select the specific tag group number to specify the tagged expression you want to use, in the format '\' followed by a tag group number 1-9. (Count the left parentheses '**(**' in the search string to determine a tagged expression number. The first (left-most) tagged expression is "**\1**" and the last is "**\9**".)

Understanding Basic VBScript Syntax

VBScript is an easy-to-learn, yet powerful scripting language. You can use VBScript to develop scripts to perform both simple and complex object-based tasks, even if you have no previous programming experience.

This section provides some basic guidelines to help you use VBScript statements to enhance your QuickTest test or function library. For more detailed information on using VBScript, you can view the VBScript documentation from the QuickTest **Help** menu (**Help > QuickTest Professional Help > VBScript Reference**).

Each VBScript statement has its own specific syntax rules. If you do not follow these rules, errors will be generated when you run the problematic step. Additionally, if you try to move to the Keyword View from the Expert View, QuickTest lists any syntax errors found in the document in the Information pane. You cannot switch to the Keyword View without fixing or eliminating the syntax errors. For more information, see “Handling VBScript Syntax Errors” on page 860.

Tip: You can check the syntax of the current document at any time by clicking the **Check Syntax** button, or by choosing **Tools** > **Check Syntax**. If a test is open, the syntax of all the actions is checked. If a function library is open, the syntax of the library script is checked.

When working in the Expert View or in a function library, you should consider the following general VBScript syntax rules and guidelines:

- **Case-sensitivity.** By default, VBScript is not case sensitive and does not differentiate between upper-case and lower-case spelling of words, for example, in variables, object and operation names, or constants.

  For example, the two statements below are identical in VBScript:

  ```
  Browser("Mercury").Page("Find a Flight:").WebList("toDay").Select "31"
  browser("mercury").page("find a flight:").weblist("today").select "31"
  ```

- **Text strings.** When you enter a value as a text string, you must add quotation marks before and after the string. For example, in the above segment of script, the names of the Web site, Web page, and edit box are all text strings surrounded by quotation marks.

  Note that the value 31 is also surrounded by quotation marks, because it is a text string that represents a number and not a numeric value.

  In the following example, only the property name (first argument) is a text string and is in quotation marks. The second argument (the value of the property) is a variable and therefore does not have quotation marks. The third argument (specifying the timeout) is a numeric value, which also does not need quotation marks.

  ```
  Browser("Mercury").Page("Find a Flight:").WaitProperty("items count",
      Total_Items, 2000)
  ```

- **Variables.** You can specify variables to store strings, integers, arrays and objects. Using variables helps to make your script more readable and flexible. For more information, see “Using Variables” on page 856.

- **Parentheses.** To achieve the desired result and to avoid errors, it is important that you use parentheses () correctly in your statements. For more information, see “Using Parentheses” on page 857.

- **Indentation.** You can indent or outdent your script to reflect the logical structure and nesting of the statements. For more information, see "Formatting VB Script Text" on page 859.
- **Comments.** You can add comments to your statements using an apostrophe ('), either at the beginning of a separate line, or at the end of a statement. It is recommended that you add comments wherever possible, to make your scripts easier to understand and maintain. For more information, see "Formatting VB Script Text" on page 859, and "Inserting Comments" on page 877.
- **Spaces.** You can add extra blank spaces to your script to improve clarity. These spaces are ignored by VBScript.

  For more information on using specific VBScript statements to enhance your tests or function libraries, see "Using Comments, Control-Flow, and Other VBScript Statements" on page 876.

Using Variables

You can specify variables to store test objects or simple values in your test or function library. When using a variable for a test object, you can use the variable instead of the entire object hierarchy in other statements. Using variables in this way makes your statements easier to read and to maintain.

To specify a variable to store an object, use the **Set** statement, with the following syntax:

Set *ObjectVar = ObjectHierarchy*

In the example below, the **Set** statement specifies the variable **UserEditBox** to store the full **Browser** > **Page** > **WebEdit** object hierarchy for the **username** edit box. The **Set** method then enters the value John into the **username** edit box, using the **UserEditBox** variable:

```
Set UserEditBox = Browser("Mercury Tours").Page("Mercury Tours").
    WebEdit("username")
UserEditBox.Set "John"
```

Note: Do not use the **Set** statement to specify a variable containing a simple value (such as a string or a number). The example below shows how to define a variable for a simple value:

```
MyVar = Browser("Mercury Tours").Page("Mercury Tours").
   WebEdit("username").GetTOProperty("type")
```

You can also use the **Dim** statement to declare variables of other types, including strings, integers, and arrays. This statement is not mandatory, but you can use it to improve the structure of your test or function library. In the following example, the **Dim** statement is used to declare the passengers variable, which can then be used in different statements within the current action or function library:

```
Dim passengers
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").GetROProperty("value")
```

Using Parentheses

When programming in VBScript, it is important that you follow the rules for using or not using parentheses () in your statements.

You must use parentheses around method arguments if you are calling a method that returns a value and you are using the return value.

For example, use parentheses around method arguments if you are returning a value to a variable, if you are using the method in an **If** statement, or if you are using the **Call** keyword to call an action or function. You also need to add parentheses around the name of a checkpoint if you want to retrieve its return value.

Tip: If you receive an **Expected end of statement** error message when running a step in your test or function library, it may indicate that you need to add parentheses around the arguments of the step's method.

Following are several examples showing when to use or not use parentheses.

The following example requires parentheses around the method arguments for the **ChildItem** method because it returns a value to a variable.

```
Set WebEditObj = Browser("Mercury Tours").Page("Method of Payment").
    WebTable("FirstName").ChildItem (8, 2, "WebEdit", 0)
WebEditObj.Set "Example"
```

The following example requires parentheses around the method arguments because **Call** is being used.

```
Call RunAction("BookFlight", oneIteration)
```

or

```
Call MyFunction("Hello World")
...
...
```

The following example requires parentheses around the **WaitProperty** method arguments because the method is used in an **If** statement.

```
If Browser("index").Page("index").Link("All kind of").
    WaitProperty("attribute/readyState", "complete", 4) Then
    Browser("index").Page("index").Link("All kind of").Click
End If
```

The following example requires parentheses around the **Check** method arguments, since it returns the value of the checkpoint.

```
a = Browser("MyBrowser").Page("MyPage").Check (CheckPoint("MyProperty"))
```

The following example does not require parentheses around the **Click** method arguments because it does not return a value.

```
Browser("Mercury Tours").Page("Method of Payment").WebTable("FirstName").
    Click 3,4
```

Formatting VB Script Text

When working in the Expert View or in a function library, it is important to follow accepted VBScript practices for comments and indentation.

Use comments to explain sections of a script. This improves readability and make tests and function libraries easier to maintain and update. For more information, see "Inserting Comments" on page 877.

Use indentation to reflect the logical structure and nesting of your statements.

- **Adding Comments.** You can add comments to your statements by adding an apostrophe ('), either at the beginning of a separate line, or at the end of a statement.

Tips:

- You can comment a statement by clicking anywhere in the statement and clicking the **Comment Block** button.
- You can comment a selected block of text by clicking the **Comment Block** button, or by choosing **Edit** > **Advanced** > **Comment Block**. Each line in the block will be preceded by an apostrophe.

- **Removing Comments.** You can remove comments from your statements by deleting the apostrophe ('), either at the beginning of a separate line, or at the end of a statement.

Tip: You can remove the comments from a selected block or line of text by clicking the **Uncomment Block** button, or by choosing **Edit** > **Advanced** > **Uncomment Block**.

➤ **Indenting Statements.** You can indent your statements by selecting the text and choosing **Edit** > **Advanced** > **Indent** or by press the TAB key. The text is indented according to the tab spacing selected in the Editor Options dialog box, as described in "Customizing Editor Behavior" on page 897.

Note: The **Indent selected text when using the Tab key** check box must be selected in the Editor Options dialog box, otherwise pressing the TAB key will delete the selected text.

➤ **Outdenting Statements.** You can outdent your statements by selecting **Edit** > **Advanced** > **Outdent** or by deleting the space at the beginning of the statements.

For more detailed information on formatting in VBScript, you can view the VBScript documentation from the QuickTest **Help** menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).

Handling VBScript Syntax Errors

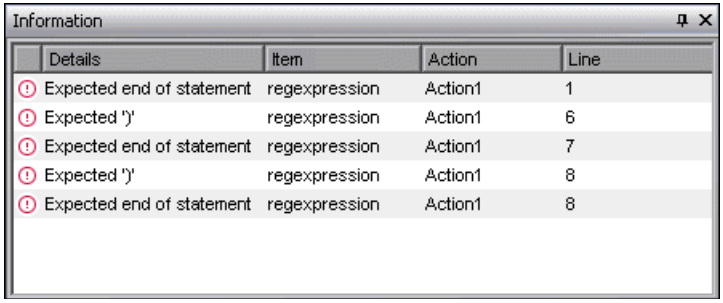
When you select the Keyword View tab from the Expert View, QuickTest attempts to display the updated information in the Keyword View. If a new or updated VBScript statement contains syntax errors, the text **Error** flashes in red at the right of the status bar, and an error message is displayed in the status bar informing you that you should view the Information pane for information about syntax errors in the script. QuickTest is unable to display the document in the Keyword View until you have fixed all the syntax errors.

You can view a description of each of the VBScript errors in the VBScript Reference. For more information, select **Help** > **QuickTest Professional Help** > **VBScript Reference** > **VBScript** > **Reference** > **Errors** > **VBScript Syntax Errors**.

Tips:

- You can check the syntax of the current document at any time by clicking the **Check Syntax** button, or by choosing **Tools** > **Check Syntax**. If a test is open, the syntax of all the actions is checked. If a function library is open, the syntax of the library script is checked.
- The Microsoft VBScript Language Reference defines VBScript syntax errors as: "errors that result when the structure of one of your VBScript statements violates one or more of the grammatical rules of the VBScript scripting language." To learn about working with VBScript, you can view the VBScript Reference from the QuickTest **Help** menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).

The Information pane lists the syntax errors found in your document, and enables you to locate each syntax error so that you can correct it.



The Information pane shows the following information for each syntax error:

Pane Element	Description
Details	The description of the syntax error. For example, if you opened a conditional block with an **If** statement but did not close it with an **End If** statement, the description is Expected 'End If'. **Note:** In certain cases, QuickTest is unable to identify the exact error and displays a number of possible error conditions, for example: Expected 'End Sub', or 'End Function', or 'End Property'. Check the statement at the specified line to clarify which error is relevant in your case.
Item	The name of the test or function library containing the problematic statement.
Action	The name of the action containing the problematic statement. This column is not relevant for function libraries that are associated with business components (via application areas).
Line	The line containing the syntax error. Lines are numbered from the beginning of each action or function library.

Using the Information Pane

- Move the pointer over the description of a syntax error to display the currently incorrect syntax.
- To navigate to the line containing a specific syntax error, double-click the syntax error in the Information pane.
- You can resize the columns in the Information pane to make the information more readable by dragging the column headers.
- You can sort the details in the Information pane in ascending or descending order by clicking the column header.
- You can press F1 on an error in the Information pane to display information about VBScript syntax errors.

Using Programmatic Descriptions

When QuickTest learns an object in your application, it adds the appropriate test object to the object repository. After the object exists in the object repository, you can add statements in the Expert View to perform additional operations on that object. To add these statements, you usually enter the name (not case sensitive) of each of the objects in the object's hierarchy as the object description, and then add the appropriate operation.

For example, in the statement below, username is the name of an edit box. The edit box is located on a page with the name Mercury Tours, and the page exists in a browser with the name Mercury Tours.

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username")
```

Because each object in the object repository has a unique name, the object name is all you need to specify. During the run session, QuickTest finds the object in the object repository based on its name and parent objects, and uses the stored test object description for that test object to identify the object in your application.

You can also instruct QuickTest to perform operations on objects without referring to the object repository or to the object's name. To do this, you provide QuickTest with a list of properties and values that QuickTest can use to identify the object or objects on which you want to perform an operation.

Such a **programmatic description** can be very useful if you want to perform an operation on an object that is not stored in the object repository. You can also use programmatic descriptions to perform the same operation on several objects with certain identical properties, or to perform an operation on an object whose properties match a description that you determine dynamically during the run session.

In the Test Results, square brackets around a test object name indicate that the test object was created dynamically during the run session using a programmatic description or the **ChildObjects** method.

For example, suppose you are testing a Web site that generates a list of potential employers based on biographical information you provide, and offers to send your resume to the employer names you select from the list. You want your test to select all the employers displayed in the list, but when you design your test, you do not know how many check boxes will be displayed on the page, and you cannot, of course, know the exact object description of each check box. In this situation, you can use a programmatic description to instruct QuickTest to perform a Set "ON" method for all objects that fit the description: HTML TAG = input, TYPE = check box.

There are two types of programmatic descriptions:

- **Static.** You list the set of properties and values that describe the object directly in a VBScript statement.
- **Dynamic.** You add a collection of properties and values to a Description object, and then enter the Description object name in the statement.

Using the **Static** type to enter programmatic descriptions directly into your statements may be easier for basic object description needs. However, in most cases, using the **Dynamic** type provides more power, efficiency, and flexibility.

Entering Programmatic Descriptions Directly into Statements

You can describe an object directly in a statement by specifying **property:=value** pairs describing the object instead of specifying an object's name.

The general syntax is:

TestObject("PropertyName1:=PropertyValue1", "..." ,
"PropertyNameX:=PropertyValueX")

TestObject. The test object class.

PropertyName:=PropertyValue. The identification property and its value. Each **property:=value** pair should be separated by commas and quotation marks.

Note that you can enter a variable name as the property value if you want to find an object based on property values you retrieve during a run session. For example:

```
MyVar="some text string"
Browser("Hello").Page("Hello").Webtable("table").Webedit("name:=" & MyVar)
```

Note: QuickTest evaluates all property values in programmatic descriptions as regular expressions. Therefore, if you want to enter a value that contains a special regular expression character (such as *, ?, or +), use the \ (backslash) character to instruct QuickTest to treat the special characters as literal characters. For more information on regular expressions, see "Understanding and Using Regular Expressions" on page 762.

The statement below specifies a WebEdit test object in the Mercury Tours page with the Name author and an index of 3. During the run session, QuickTest finds the WebEdit object with matching property values and enters the text Mark Twain.

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("Name:=Author",
    "Index:=3").Set "Mark Twain"
```

Note: When using programmatic descriptions from a specific point within a test object hierarchy, you must continue to use programmatic descriptions from that point onward within the same statement. If you specify a test object by its object repository name after other objects in the hierarchy have been specified using programmatic descriptions, QuickTest cannot identify the object.

For example, you can use the following statement since it uses programmatic descriptions throughout the entire test object hierarchy:

```
Browser("Title:=Mercury Tours").Page("Title:=Mercury Tours").
  WebEdit("Name:=Author", "Index:=3").Set "Mark Twain"
```

You can also use the statement below, since it uses programmatic descriptions from a certain point in the description (starting from the Page object description):

```
Browser("Mercury Tours").Page("Title:=Mercury Tours").
  WebEdit("Name:=Author", "Index:=3").Set "Mark Twain"
```

However, you cannot use the following statement, since it uses programmatic descriptions for the Browser and Page objects but then attempts to use an object repository name for the WebEdit test object:

```
Browser("Title:=Mercury Tours").Page("Title:=Mercury Tours").
  WebEdit("Author").Set "Mark Twain"
```

QuickTest tries to locate the WebEdit object based on its name, but cannot locate it in the repository because the parent objects were specified using programmatic descriptions.

For more information on working with test objects, see Chapter 5, “Managing Test Objects in Object Repositories.”

If you want to use the same programmatic description several times in one test or function library, you may want to assign the object you create to a variable.

For example, instead of entering:

```
Window("Text:=Myfile.txt - Notepad").Move 50, 50
Window("Text:=Myfile.txt - Notepad").WinEdit("AttachedText:=Find what:").
    Set "hello"
Window("Text:=Myfile.txt - Notepad").WinButton("Caption:=Find next").Click
```

You can enter:

```
Set MyWin = Window("Text:=Myfile.txt - Notepad")
MyWin.Move 50, 50
MyWin.WinEdit("AttachedText:=Find what:").Set "hello"
MyWin.WinButton("Caption:=Find next").Click
```

Alternatively, you can use a **With** statement:

```
With Window("Text:=Myfile.txt - Notepad")
    .Move 50, 50
    .WinEdit("AttachedText:=Find what:").Set "hello"
    .WinButton("Caption:=Find next").Click
End With
```

For more information on the **With** statement, see “With Statement” on page 884.

Using Description Objects for Programmatic Descriptions

You can use the **Description** object to return a **Properties** collection object containing a set of **Property** objects. A **Property** object consists of a property name and value. You can then specify the returned **Properties** collection in place of an object name in a statement. (Each property object contains a property name and value pair.)

Note: By default, the value of all **Property** objects added to a **Properties** collection are treated as regular expressions. Therefore, if you want to enter a value that contains a special regular expression character (such as *****, ?, +), use the \ (backslash) character to instruct QuickTest to treat the special characters as literal characters. For more information on regular expressions, see "Understanding and Using Regular Expressions" on page 762.

You can set the **RegularExpression** property to False to specify a value as a literal value for a specific **Property** object in the collection. For more information, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

To create the **Properties** collection, you enter a **Description.Create** statement using the following syntax:

Set *MyDescription* = **Description.Create()**

After you have created a **Properties** object (such as MyDescription in the example above), you can enter statements to add, edit, remove, and retrieve properties and values to or from the **Properties** object during the run session. This enables you to determine which, and how many properties to include in the object description in a dynamic way during the run session.

After you fill the **Properties** collection with a set of Property objects (properties and values), you can specify the **Properties** object in place of an object name in a test statement.

For example, instead of entering:

```
Window("Error").WinButton("text:=OK", "width:=50").Click
```

you can enter:

```
Set MyDescription = Description.Create()
MyDescription("text").Value = "OK"
MyDescription("width").Value = 50
Window("Error").WinButton(MyDescription).Click
```

Note: When using programmatic descriptions from a specific point within a test object hierarchy, you must continue to use programmatic descriptions from that point onward within the same statement. If you specify a test object by its object repository name after other objects in the hierarchy have been described using programmatic descriptions, QuickTest cannot identify the object.

For example, you can use `Browser(Desc1).Page(Desc1).Link(desc3)`, since it uses programmatic descriptions throughout the entire test object hierarchy.

You can also use `Browser("Index").Page(Desc1).Link(desc3)`, since it uses programmatic descriptions from a certain point in the description (starting from the Page object description).

However, you cannot use `Browser(Desc1).Page(Desc1).Link("Example1")`, since it uses programmatic descriptions for the Browser and Page objects but then attempts to use an object repository name for the Link test object (QuickTest tries to locate the Link object based on its name, but cannot locate it in the repository because the parent objects were specified using programmatic descriptions).

When working with **Properties** objects, you can use variable names for the properties or values to generate the object description based on properties and values you retrieve during a run session.

You can create several **Properties** objects in your test if you want to use programmatic descriptions for several objects.

For more information on the **Description** and **Properties** objects and their associated methods, see the *HP QuickTest Professional Object Model Reference*.

Retrieving Child Objects

You can use the **ChildObjects** method to retrieve all objects located inside a specified parent object, or only those child objects that fit a certain programmatic description. To retrieve this subset of child objects, you first create a description object and add the set of properties and values that you want your child object collection to match using the **Description** object.

Note: You must use the **Description** object to create the programmatic description for the **ChildObjects** description argument. You cannot enter the programmatic description directly into the argument using the **property:=value** syntax.

After you have "built" a description in your description object, use the following syntax to retrieve child objects that match the description:

Set *MySubSet=TestObject*.**ChildObjects(***MyDescription***)**

For example, the statements below instruct QuickTest to select all of the check boxes on the Itinerary Web page:

```
Set MyDescription = Description.Create()
MyDescription("html tag").Value = "INPUT"
MyDescription("type").Value = "checkbox"

Set Checkboxes =
Browser("Itinerary").Page("Itinerary").ChildObjects(MyDescription)
NoOfChildObjs = Checkboxes.Count
For Counter=0 to NoOfChildObjs-1
    Checkboxes(Counter).Set "ON"
Next
```

In the Test Results, square brackets around a test object name indicate that the test object was created dynamically during the run session using the **ChildObjects** method or a programmatic description.

For more information on the **ChildObjects** method, see the *HP QuickTest Professional Object Model Reference*.

Using the Index Property in Programmatic Descriptions

The index property can sometimes be a useful identification property for uniquely identifying an object. The **index** identification property identifies an object based on the order in which it appears within the source code, where the first occurrence is 0.

Index property values are object-specific. Thus, if you use an index value of 3 to describe a WebEdit test object, QuickTest searches for the fourth WebEdit object in the page.

If you use an index value of 3 to describe a WebElement object, however, QuickTest searches for the fourth Web object on the page regardless of the type, because the WebElement object applies to all Web objects.

For example, suppose you have a page with the following objects:

- An image with the name Apple
- An image with the name UserName
- A WebEdit object with the name UserName
- An image with the name Password
- A WebEdit object with the name Password

The description below refers to the third item in the list above, as it is the first WebEdit object on the page with the name UserName:

```
WebEdit("Name:=UserName", "Index:=0")
```

The following description, however, refers to the second item in the list above, as that is the first object of any type (WebElement) with the name UserName:

```
WebElement("Name:=UserName", "Index:=0")
```

Note: If there is only one object, using index=0 will not retrieve it. You should not include the **index** property in the object description.

Performing Programmatic Description Checks

You can compare the run-time value of a specified object property with the expected value of that property using either programmatic descriptions or user-defined functions.

Programmatic description checks are useful in cases in which you cannot apply a regular checkpoint, for example, if the object whose properties you want to check is not stored in an object repository. You can then write the results of the check to the Test Results report.

For example, suppose you want to check the run-time value of a Web button. You can use the **GetROProperty** or **Exist** operations to retrieve the run-time value of an object or to verify whether the object exists at that point in the run session.

The following examples illustrate how to use programmatic descriptions to check whether the **Continue** Web button is disabled during a run session.

Using the **GetROProperty** operation:

```
ActualDisabledVal =
Browser(micClass:="Browser").Page(micClass:="Page").WebButton
    (alt:="Continue").GetROProperty("disabled")
```

Using the **Exist** operation:

```
While Not Browser(micClass:="Browser").Page(micClass:="Page").WebButton
    (alt:="Continue").Exist(30)
Wend
```

By adding **Report.ReportEvent** statements, you can instruct QuickTest to send the results of a check to the Test Results:

```
If ActualDisabledVal = True Then
Reporter.ReportEvent micPass, "CheckContinueButton = PASS", "The
Continue
    button is disabled, as expected."
Else
Reporter.ReportEvent micFail, "CheckContinueButton = FAIL", "The Continue
    button is enabled, even though it should be disabled."
```

You can also create and use user-defined functions to check whether your application is functioning as expected. The following example illustrates a function that checks whether an object is disabled and returns **True** if the object is disabled:

```
'@Description Checks whether the specified test object is disabled
'@Documentation Check whether the <Test object name> <test object type> is
enabled.
Public Function VerifyDisabled (obj)
    Dim enable_property
    ' Get the disabled property from the test object
    enable_property = obj.GetROProperty("disabled")
    If enable_property = 1 Then ' The value is True (1)—the object is disabled
        Reporter.ReportEvent micPass, "VerifyDisabled Succeeded", "The test
object is disabled, as expected."
        VerifyDisabled = True
    Else
        Reporter.ReportEvent micFail, "VerifyDisabled Failed", "The test object is
enabled, although it should be disabled."
        VerifyDisabled = False
    End If
End Function
```

Note: For information on using the **GetROProperty** operation, see "Retrieving Native Properties" on page 888. For information on using **While...Wend** statements, see "While...Wend Statement" on page 882. For information on specific test objects, operations, and properties, see the *HP QuickTest Professional Object Model Reference*.

Running and Closing Applications Programmatically

In addition to using the Record and Run Settings dialog box to instruct QuickTest to open a new application when a test run begins, or manually opening the application you want to test, you can insert statements into your test that open and close the applications you want to test.

You can run any application from a specified location using a **SystemUtil.Run** statement. This is especially useful if your test includes more than one application, and you selected the **Record and run test on any application** check box in the Record and Run Settings dialog box. You can specify an application and pass any supported parameters, or you can specify a file name and the associated application starts with the specified file open.

You can close most applications using the **Close** method. You can also use **SystemUtil** statements to close applications. For more information, see the *HP QuickTest Professional Object Model Reference*.

For example, you could use the following statements to open a file named **type.txt** in the default text application (Notepad), type happy days, save the file using shortcut keys, and then close the application:

```
SystemUtil.Run "C:\type.txt", "","",""
Window("Text:=type.txt - Notepad").Type "happy days"
Window("Text:=type.txt - Notepad").Type micAltDwn & "F" & micAltUp
Window("Text:=type.txt - Notepad").Type micLShiftDwn & "S" & micLShiftUp
Window("Text:=type.txt - Notepad").Close
```

Notes:

- When you specify an application to open using the Record and Run Settings dialog box, QuickTest does not add a **SystemUtil.Run** statement to your test.
- The **InvokeApplication** method can open only executable files and is used primarily for backward compatibility.

For more information, see the *HP QuickTest Professional Object Model Reference*.

Using Comments, Control-Flow, and Other VBScript Statements

QuickTest enables you to incorporate decision-making into your test or function library by adding conditional statements that control the logical flow of your test or function library. In addition, you can define messages in your test that QuickTest sends to your test results. To improve the readability of your tests and function libraries, you can also add comments to them.

For information on how to use these programming concepts in the Keyword View, see Chapter 28, "Adding Steps Containing Programming Logic."

Note: The **VBScript Reference** (available from **Help** > **QuickTest Professional Help**) contains Microsoft VBScript documentation, including VBScript, Script Runtime, and Windows Script Host.

Inserting Comments

A comment is a line or part of a line in a script that is preceded by an apostrophe ('). When you run a test or a function in a function library, QuickTest does not process the comments. Use comments to explain sections of a script to improve readability and to make tests and function libraries easier to update.

The following example shows how a comment describes the purpose of the statement below it:

```
'Sets the word "mercury" into the "username" edit box.
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").
    Set "mercury"
```

By default, comments are displayed in green in the Expert View and in function libraries. You can customize the appearance of comments in the Editor Options dialog box. For more information, see "Customizing Element Appearance" on page 900.

Tips:

- You can comment a block of text by choosing **Edit** > **Advanced** > **Comment Block** or by clicking the **Comment Block** button.

- To remove the comment, select **Edit** > **Advanced** > **Uncomment Block** or click the **Uncomment Block** button.

Note: You can also add a comment line using the VBScript **Rem** statement. For more information, see the Microsoft VBScript Language Reference (select **Help** > **QuickTest Professional Help** > **VBScript Reference** > **VBScript**).

Performing Calculations

You can write statements that perform simple calculations using mathematical operators. For example, you can use a multiplication operator to multiply the values displayed in two text boxes in your Web site. VBScript supports the following mathematical operators:

Operator	Description
+	addition
–	subtraction
–	negation (a negative number)
*	multiplication
/	division
^	exponent

In the following example, the multiplication operator is used to calculate the maximum luggage weight of the passengers at 100 pounds each:

```
'Retrieves the number of passengers from the edit box using the GetROProperty method

passenger = Browser ("Mercury_Tours").Page ("Find_Flights").
    WebEdit("numPassengers").GetROProperty("value")

'Multiplies the number of passengers by 100

weight = passenger * 100

'Inserts the maximum weight into a message box.

msgbox("The maximum weight for the party is "& weight &"pounds.")
```

For...Next Statement

A **For...Next** loop instructs QuickTest to perform one or more statements a specified number of times. It has the following syntax:

For *counter* = *start* **to** *end* **[Step** *step***]**
    statement
Next

Item	Description
counter	The variable used as a counter for the number of iterations.
start	The start number of the counter.
end	The last number of the counter.
step	The number to increment at the end of each loop. **Default = 1**. **Optional**.
statement	A statement, or series of statements, to be performed during the loop.

In the following example, QuickTest calculates the factorial value of the number of passengers using the **For** statement:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numPassengers").GetROProperty("value")
total = 1
For i=1 To passengers
    total = total * i
Next
MsgBox "!" & passengers & "=" & total
```

For...Each Statement

A **For...Each** loop instructs QuickTest to perform one or more statements for each element in an array or an object collection. It has the following syntax:

For Each *item* **In** *array*
    statement
Next

Item	Description
item	A variable representing the element in the array.
array	The name of the array.
statement	A statement, or series of statements, to be performed during the loop.

The following example uses a **For...Each** loop to display each of the values in an array:

```
MyArray = Array("one","two","three","four","five")
For Each element In MyArray
    msgbox element
Next
```

Do...Loop Statement

The **Do...Loop** statement instructs QuickTest to perform a statement or series of statements while a condition is true or until a condition becomes true. It has the following syntax:

Do [{while} {until} c*ondition*]
    statement
Loop

Item	Description
condition	A condition to be fulfilled.
statement	A statement or series of statements to be performed during the loop.

In the following example, QuickTest calculates the factorial value of the number of passengers using the **Do...Loop**:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numPassengers").GetROProperty("value")
total = 1
i = 1
Do while i <= passengers
     total = total * i
     i = i + 1
Loop
MsgBox "!" & passengers & "=" & total
```

While...Wend Statement

A **While...Wend** statement instructs QuickTest to perform a statement or series of statements while a condition is true. It has the following syntax:

While *condition*
    statement
Wend

Item	Description
condition	A condition to be fulfilled.
statement	A statement or series of statements to be executed during the loop.

In the following example, QuickTest performs a loop using the **While** statement while the number of passengers is fewer than ten. Within each loop, QuickTest increments the number of passengers by one:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").GetROProperty("value")
While passengers < 10
    passengers = passengers + 1
Wend

msgbox("The number of passengers in the party is " & passengers)
```

If...Then...Else Statement

The **If...Then...Else** statement instructs QuickTest to perform a statement or a series of statements based on specified conditions. If a condition is not fulfilled, the next **ElseIf** condition or **Else** statement is examined. It has the following syntax:

If *condition* **Then**
    statement
ElseIf *condition2* **Then**
    statement
Else
    statement
End If

Item	Description
condition	Condition to be fulfilled.
statement	Statement to be perform.

In the following example, if the number of passengers is fewer than four, QuickTest closes the browser:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").GetROProperty("value")
If (passengers < 4) Then
    Browser("Mercury Tours").Close
Else
    Browser("Mercury Tours").Page("Find Flights").Image("continue").Click 69,5
End If
```

The following example uses **If**, **ElseIf**, and **Else** statements to check whether a value is equal to 1, 2, or a different value:

```
value = 2
If value = 1 Then
  msgbox "one"
ElseIf value = 2 Then
  msgbox "two"
Else
  msgbox "not one or two"
End If
```

With Statement

With statements make your script more concise and easier to read and write or edit by grouping consecutive statements with the same parent hierarchy.

Note: When running a **With** statement, QuickTest identifies the object in the application before running the first statement, but does not re-identify it before running each statement. This can affect the running of your test if the object referenced by the **With** statement is refreshed, redrawn, or changed in some way in the application while running the **With** statement. To instruct QuickTest to re-identify the object in the application before running the next statement, add a statement that calls the **RefreshObject** test object operation. For more information on the **RefreshObject** operation, see the *HP QuickTest Professional Object Model Reference*.

The **With** statement has the following syntax:

With *object*
  statements
End With

Item	Description
object	An object or a function that returns an object.
statements	One or more statements to be performed on an object.

For example, you could replace this script:

```
Window("Flight Reservation").WinComboBox("Fly From:").Select "London"
Window("Flight Reservation").WinComboBox("Fly To:").Select "Los Angeles"
Window("Flight Reservation").WinButton("FLIGHT").Click
Window("Flight Reservation").Dialog("Flights Table").WinList("From").
    Select "19097   LON "
Window("Flight Reservation").Dialog("Flights Table").WinButton("OK").Click
```

with the following:

```
With Window("Flight Reservation")
    .WinComboBox("Fly From:").Select "London"
    .WinComboBox("Fly To:").Select "Los Angeles"
    .WinButton("FLIGHT").Click
    With .Dialog("Flights Table")
        .WinList("From").Select "19097   LON "
        .WinButton("OK").Click
    End With 'Dialog("Flights Table")
End With 'Window("Flight Reservation")
```

Note that entering **With** statements in the Expert View does not affect the Keyword View in any way.

Note: In addition to entering **With** statements manually, you can also instruct QuickTest to automatically generate **With** statements as you record or to generate **With** statements for an existing test. For more information, see "Generating With Statements for Your Test" on page 806.

Retrieving and Setting Identification Property Values

Identification properties are the set of properties defined by QuickTest for each object. You can set and retrieve a test object's identification property values, and you can retrieve the values of identification properties from a run-time object.

When you run your test or function, QuickTest creates a temporary version of the test object that is stored in the test object repository. You can use the **GetTOProperty**, **GetTOProperties**, and **SetTOProperty** methods in your test or function library to set and retrieve the identification property values of the test object.

The **GetTOProperty** and **GetTOProperties** methods enable you to retrieve a specific property value or all the properties and values that QuickTest uses to identify an object.

The **SetTOProperty** method enables you to modify a property value that QuickTest uses to identify an object.

Note: Because QuickTest refers to the temporary version of the test object during the run session, any changes you make using the **SetTOProperty** method apply only during the course of the run session, and do not affect the values stored in the test object repository.

For example, the following statements would set the **Submit** button's name value to my button, and then retrieve the value my button to the **ButtonName** variable:

```
Browser("QA Home Page").Page("QA Home Page").
    WebButton("Submit").SetTOProperty "Name", "my button"

ButtonName=Browser("QA Home Page").Page("QA Home Page").
    WebButton("Submit").GetTOProperty("Name")
```

You use the **GetROProperty** method to retrieve the current value of an identification property from a run-time object in your application.

For example, you can retrieve the target value of a link during the run session as follows:

```
link_href = Browser("HP Technologies").Page("HP Technologies").
    Link("Jobs").GetROProperty("href")
```

Tip: If you do not know the identification properties of objects in your application, you can view them using the Object Spy. For information on the Object Spy, see Chapter 3, "Understanding the Test Object Model."

For a list and description of identification properties supported by each object, and for more information on the **GetROProperty**, **GetTOProperty**, **GetTOProperties**, and **SetTOProperty** methods, see the *HP QuickTest Professional Object Model Reference*.

Accessing Native Properties and Operations

If the test object operations and identification properties available for a particular test object do not provide the functionality you need, you can access the native operations and properties of any run-time object in your application using the **Object** property.

You can use the statement completion feature with object properties to view a list of the available native operations and properties of an object. For more information on the statement completion option, see "Generating Statements in the Expert View or in a Function Library" on page 833.

Tip: If the object is a Web object, you can also reference its native properties in programmatic descriptions using the attribute/property notation. For more information, see "Accessing User-Defined Properties of Web Objects" on page 888.

Retrieving Native Properties

You can use the **Object** property to access the native properties of any run-time object. For example, you can retrieve the current value of the ActiveX calendar's internal **Day** property as follows:

```
Dim MyDay
Set MyDay=
Browser("index").Page("Untitled").ActiveX("MSCAL.Calendar.7").Object.Day
```

For more information on the **Object** property, see the *HP QuickTest Professional Object Model Reference*.

Activating Native Operations

You can use the **Object** property to activate the internal operations of any run-time object. For example, you can activate the native **focus** method of the edit box as follows:

```
Dim MyWebEdit
Set MyWebEdit=Browser("Mercury Tours").Page("Mercury Tours").
    WebEdit("username").Object
MyWebEdit.focus
```

For more information on the **Object** property, see the *HP QuickTest Professional Object Model Reference*.

Accessing User-Defined Properties of Web Objects

You can use the **attribute/<property name>** notation to access native properties of Web objects and use these properties to identify such objects with programmatic descriptions.

For example, suppose a Web page has the same company logo image in two places on the page:

```
<IMG src="logo.gif" LogoID="122">
<IMG src="logo.gif" LogoID="123">
```

You could identify the image that you want to click using a programmatic description by including the user-defined property LogoID in the description as follows:

```
Browser("Mercury Tours").Page("Find Flights").Image("src:=logo.gif",
    "attribute/LogoID:=123").Click 68, 12
```

For more information on programmatic descriptions, see "Using Programmatic Descriptions" on page 863.

Running DOS Commands

You can run standard DOS commands in your QuickTest test or function using the VBScript Windows Scripting Host Shell object (WSCript.shell). For example, you can open a DOS command window, change the path to C:\, and run the **DIR** command using the following statements:

```
Dim oShell
Set oShell = CreateObject ("WSCript.shell")
oShell.run "cmd /K CD C:\ & Dir"
Set oShell = Nothing
```

For more information, see the Microsoft VBScript Language Reference (select **Help** > **QuickTest Professional Help** > **VBScript Reference** > **VBScript**).

Enhancing Your Tests and Function Libraries Using the Windows API

Using the Windows API, you can extend testing abilities and add usability and flexibility to your tests and function libraries. The Windows operating system provides a large number of functions to help you control and manage Windows operations. You can use these functions to obtain additional functionality.

The Windows API is documented in the Microsoft MSDN Web site, which can be found at: http://msdn2.microsoft.com/en-us/library/Aa383750

A reference to specific API functions can be found at:
http://msdn2.microsoft.com/en-us/library/Aa383749

To use Windows API functions:

1. In MSDN, locate the function you want to use in your test or function library.
2. Read its documentation and understand all required parameters and return values.
3. Note the location of the API function. API functions are located inside Windows DLLs. The name of the DLL in which the requested function is located is usually identical to the Import Library section in the function's documentation. For example, if the documentation refers to **User32.lib**, the function is located in a DLL named **User32.dll**, typically located in your System32 library.
4. Use the QuickTest **Extern** object to declare an external function. For more information, see the *HP QuickTest Professional Object Model Reference*.

   The following example declares a call to a function called **GetForegroundWindow**, located in **user32.dll**:

   ```
   extern.declare micHwnd, "GetForegroundWindow", "user32.dll",
   "GetForegroundWindow"
   ```

5. Call the declared function, passing any required arguments, for example, `hwnd = extern.GetForegroundWindow()`.

   In this example, the foreground window's handle is retrieved. You can enhance your test or function library if the foreground window is not in the object repository or cannot be determined beforehand (for example, a window with a dynamic title). You may want to use this handle as part of a programmatic description of the window, for example:

   ```
   Window("HWND:="&hWnd).Close
   ```

In some cases, you may have to use predefined constant values as function arguments. Since these constants are not defined in the context of your test or function, you need to find their numerical value to pass them to the called function. The numerical values of these constants are usually declared in the function's header file. A reference to header files can also be found in each function's documentation under the Header section. If you have Microsoft Visual Studio installed on your computer, you can typically find header files under **X:\Program Files\Microsoft Visual Studio\VC98\Include**.

For example, the **GetWindow** API function expects to receive a numerical value that represents the relationship between the specified window and the window whose handle is to be retrieved. In the MSDN documentation, you can find the constants: GW_CHILD, GW_ENABLEDPOPUP, GW_HWNDFIRST, GW_HWNDLAST, GW_HWNDNEXT, GW_HWNDPREV and GW_HWNDPREV. If you open the **WINUSER.H** file, mentioned in the **GetWindow** documentation, you will find the following flag values:

```
/*
 * GetWindow() Constants
 */
#define GW_HWNDFIRST0
#define GW_HWNDLAST 1
#define GW_HWNDNEXT2
#define GW_HWNDPREV 3
#define GW_OWNER 4
#define GW_CHILD 5
#define GW_ENABLEDPOPUP 6
#define GW_MAX 6
```

Example

The following example retrieves a specific menu item's value in the Notepad application.

```
' Constant Values:
const MF_BYPOSITION = 1024
' API Functions Declarations
Extern.Declare micHwnd,"GetMenu","user32.dll","GetMenu",micHwnd
Extern.Declare
micInteger,"GetMenuItemCount","user32.dll","GetMenuItemCount",micHwnd
Extern.Declare
micHwnd,"GetSubMenu","user32.dll","GetSubMenu",micHwnd,micInteger
Extern.Declare
micInteger,"GetMenuString","user32.dll","GetMenuString",micHwnd,micInteger,
    micString+micByRef,micInteger,micInteger
' Notepad.exe
hwin = Window("Notepad").GetROProperty ("hwnd")' Get Window's handle
MsgBox hwin
' Use API Functions
men_hwnd = Extern.GetMenu(hwin)' Get window's main menu's handle
MsgBox men_hwnd
item_cnt = Extern.GetMenuItemCount(men_hwnd)
MsgBox item_cnt
hSubm = Extern.GetSubMenu(men_hwnd,0)
MsgBox hSubm
rc = Extern.GetMenuString(hSubm,0,value,64 ,MF_BYPOSITION)
MsgBox value
```

Choosing Which Steps to Report During the Run Session

You can use the **Report.Filter** method to determine which steps or types of steps are included in the Test Results. You can completely disable or enable reporting of steps following the statement, or you can indicate that you only want subsequent failed or failed and warning steps to be included in the report. You can also use the **Report.Filter** method to retrieve the current report mode.

The following report modes are available:

Mode	Description
0 or **rfEnableAll**	All events are displayed in the Test Results. **Default.**
1 or **rfEnableErrorsAndWarnings**	Only events with a warning or fail status are displayed in the Test Results.
2 or **rfEnableErrorsOnly**	Only events with a fail status are displayed in the Test Results.
3 or **rfDisableAll**	No events are displayed in the Test Results.

➤ To disable reporting of subsequent steps, enter the following statement:

```
Reporter.Filter = rfDisableAll
```

➤ To re-enable reporting of subsequent steps, enter:

```
Reporter.Filter = rfEnableAll
```

➤ To instruct QuickTest to include only subsequent failed steps in the Test Results, enter:

```
Reporter.Filter = rfEnableErrorsOnly
```

➤ To instruct QuickTest to include only subsequent failed or warning steps in the Test Results, enter:

```
Reporter.Filter = rfEnableErrorsAndWarnings
```

➤ To retrieve the current report mode, enter:

```
MyVar=Reporter.Filter
```

For more information, see the *HP QuickTest Professional Object Model Reference*.

30

Customizing the Expert View and Function Library Windows

You can customize the way your test is displayed when you work in the Expert View and the way functions are displayed in the function library windows. Any changes you make are applied globally to the Expert View and to all function library windows.

This chapter includes:

About Customizing the Expert View and Function Library Windows

QuickTest includes a powerful and customizable editor that enables you to modify many aspects of the Expert View and function library windows.

The Editor Options dialog box enables you to change the way scripts and function libraries are displayed in the Expert View and function library windows. You can also change the font style and size of text in your scripts and function libraries, and change the color of different elements, including comments, strings, QuickTest reserved words, operators, and numbers. For example, you can display all text strings in red.

QuickTest includes a list of default keyboard shortcuts that enable you to move the cursor, delete characters, and cut, copy, and paste information to and from the Clipboard. You can replace these shortcuts with shortcuts you prefer. For example, you could change the **Line start** command from the default HOME to ALT + HOME.

You can also modify the way your script or function library is printed using options in the Print dialog box. For more information, see "Printing a Test" on page 332 and "Printing a Function Library" on page 917.

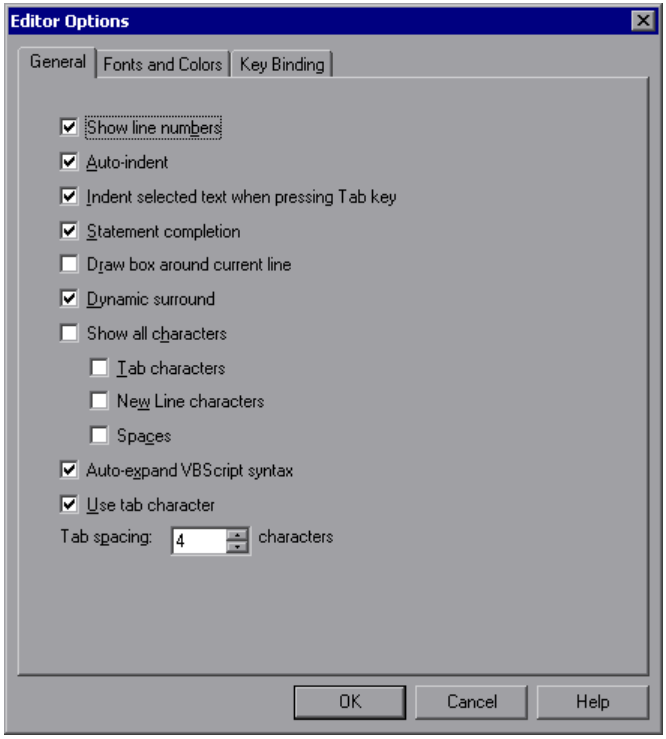
For more information on using the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows." For more information on working with function libraries, see Chapter 31, "Working with User-Defined Functions and Function Libraries."

Customizing Editor Behavior

You can customize how scripts and function libraries are displayed in the Expert View and function library windows. For example, you can show or hide character symbols, and choose to display line numbers. For more information on using the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows." For more information on working with function libraries, see Chapter 31, "Working with User-Defined Functions and Function Libraries."

To customize editor behavior:

1. When the Expert View or a function library window is active, select **Tools** > **View Options**. The Editor Options dialog box opens.
2. Click the **General** tab.



3 Select from the following options:

Options	Description
Show line numbers	Displays a line number to the left of each line in the script or function.
Auto-indent	Causes lines following an indented line to automatically begin at the same point as the previous line. You can press the HOME key on your keyboard to move the cursor back to the left margin.
Indent selected text when pressing Tab key	Pressing the TAB key indents the selected text. When this option is not enabled, pressing the Tab key replaces the selected text with a single Tab character.
Statement completion	If this option is selected, when you type in the Expert View or a function library, IntelliSense (the statement completion feature included with QuickTest) enables you to select the variable, test object, method, property, or collection for your statement from a drop-down list and view the relevant syntax. For more information on using the statement completion (IntelliSense) feature, see "Using Statement Completion (IntelliSense)" on page 833.
Draw box around current line	Displays a box around the line of the test in which the cursor is currently located.
Dynamic surround	Surrounds existing lines of code with a block structure, enabling you to dynamically expand (or collapse) block statements. For example, when you add a surrounding statement (such as if/while) before existing code, you can use the arrow keys to expand the block to include subsequent lines. These lines are then automatically indented to the correct levels.

Options	Description
Show all characters	Displays all TAB, NEW LINE, and SPACE character symbols. You can also select to display only some of these characters by selecting or clearing the relevant check boxes.

Auto-expand VBScript syntax	Automatically recognizes the first two characters of keywords and adds the relevant VBScript syntax or blocks to the script, when you type the relevant keyword.
	For example, if you enter the letters if and then enter a space at the beginning of a line in the Expert View, QuickTest automatically enters:
	If Then
	End If

Use tab character/ Tab spacing	Inserts a TAB character when the TAB key on the keyboard is used. When this option is not enabled, the specified number of space characters is inserted when you press the TAB key.

4 Click **OK** to apply the changes and close the dialog box.

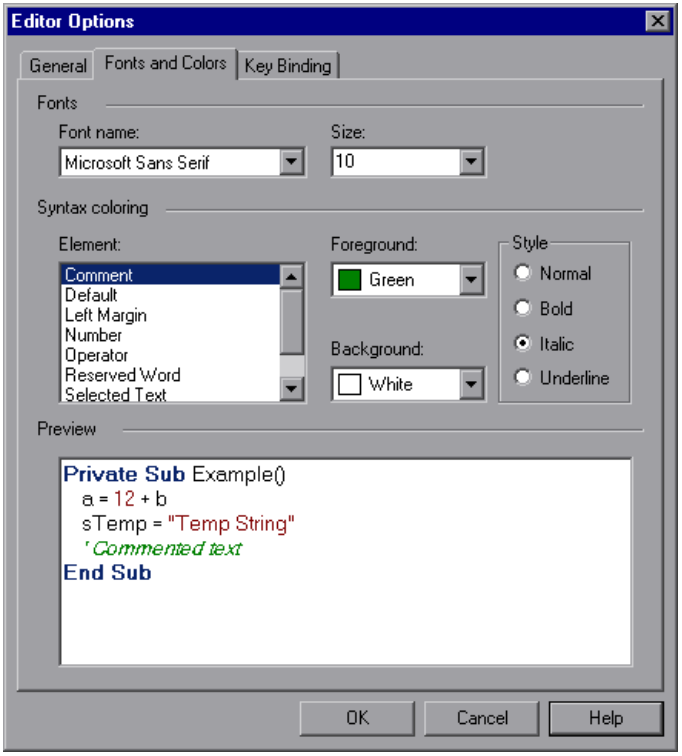
Customizing Element Appearance

QuickTest tests and function libraries contain many different elements, such as comments, strings, QuickTest and VBScript reserved words, operators, and numbers. Each element of QuickTest tests and function libraries can be displayed in a different color. You can also specify the font style and size to use for all elements. You can create your own personalized color scheme for each element. For example, all comments could be displayed as blue letters on a yellow background.

To set font and color preferences for elements:

1 When the Expert View or a function library window is active, select **Tools** > **View Options**. The Editor Options dialog box opens.

2 Click the **Fonts and Colors** tab.



3 In the **Fonts** area, select the **Font name** and **Size** that you want to use to display all elements. By default, the editor uses the Microsoft Sans Serif font, which is a Unicode font.

Note: When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your test or function library may not be correctly displayed in the Expert View or function library windows. However, the test or function library will still run in the same way, regardless of the font you choose. If you are working in an environment that is not Unicode-compatible, you may prefer to choose a fixed-width font, such as Courier, to ensure better character alignment.

---

4 Select an element from the **Element** list.

5 Choose a foreground color and a background color.

6 Choose a font style for the element (**Normal**, **Bold**, **Italic**, or **Underline**). An example of your change is displayed in the **Preview** pane at the bottom of the dialog box.

7 Repeat steps 4 to 6 for each element you want to modify.

8 Click **OK** to apply the changes and close the dialog box.

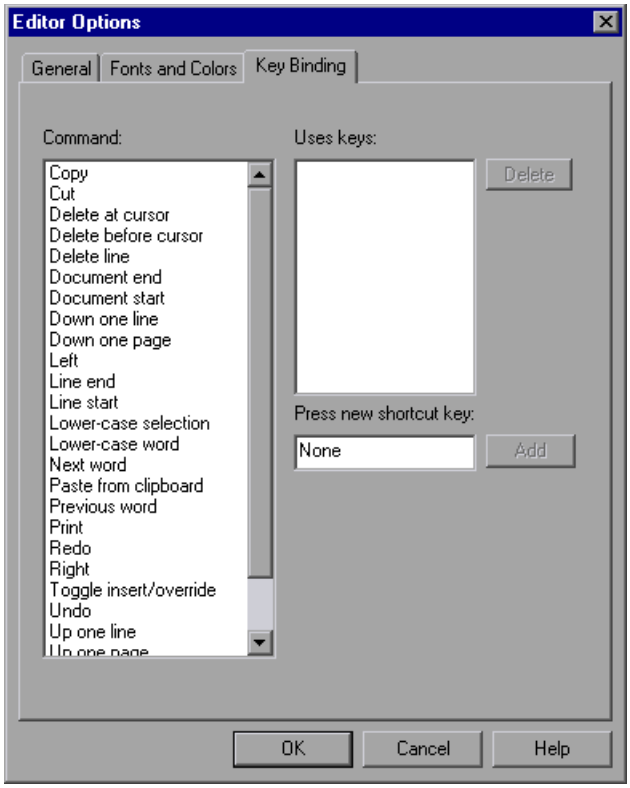
Personalizing Editing Commands

You can personalize the default keyboard shortcuts you use for editing. QuickTest includes keyboard shortcuts that let you move the cursor, delete characters, and cut, copy, or paste information to and from the Clipboard. You can replace these shortcuts with your preferred shortcuts. For example, you could change the **Line end** command from the default END to ALT + END.

Note: The default QuickTest menu shortcut keys override any key bindings that you may define. For example, if you define the Paste command key binding to be CTRL+P, it will be overridden by the default QuickTest shortcut key for opening the Print dialog box (corresponding to the **File** > **Print** option). For a complete list of QuickTest menu shortcut keys, see "Performing QuickTest Commands" on page 46.

To personalize editing commands:

1. When the Expert View or a function library window is active, select **Tools** > **View Options**. The Editor Options dialog box opens.
2. Click the **Key Binding** tab.



3. Select a command from the **Command** list.
4. Click in the **Press new shortcut key** box and then press the keys you want to use for the selected command. For example, press and hold the CTRL key while you press the number 4 key to enter CTRL+4.

5 Click **Add**.

Note: If the key combination you specify is not supported, or if it is already defined for another command, a message displays below the shortcut key box.

---

6 Repeat steps 3 to 5 for any additional commands.

7 If you want to delete a key sequence from the list, select the command in the **Command** list, then highlight the keys in the **Uses keys** list, and click **Delete**.

8 Click **OK** to apply the changes and close the dialog box.

31

Working with User-Defined Functions and Function Libraries

In addition to the test objects, methods, and built-in functions supported by the QuickTest Test Object Model, you can define your own function libraries containing VBScript functions, subroutines, modules, and so forth, and then call their functions from your test.

This chapter includes:

About Working with User-Defined Functions and Function Libraries

If you have segments of code that you need to use several times in your tests, you may want to create a user-defined function. A user-defined function encapsulates an activity (or a group of steps that require programming) into a keyword (also called an operation). By using user-defined functions, your tests are shorter, and easier to design, read, and maintain. You can then call user-defined functions from an action by inserting the relevant keywords (or operations) into that action.

You can register a user-defined function as a method for a QuickTest test object. A registered method can either override the functionality of an existing test object method for the duration of a run session, or be registered as a new method for a test object class. For more information on registering user-defined functions, see "Using the Function Definition Generator" on page 923 and "Registering User-Defined Functions as Test Object Methods" on page 939.

Note: When you create a user-defined function, do not give it the same name as a built-in function (for example, GetLastError, MsgBox, or Print). Built-in functions take priority over user-defined functions, so if you call a user-defined function that has the same name as a built-in function, the built-in function is called instead. For a list of built-in functions, see the **Built-in functions** list in the Step Generator (**Insert > Step Generator**).

Using QuickTest, you can define and store your user-defined functions either in a function library (saved as a **.qfl** file, by default) or directly in an action within a test. A function library is a Visual Basic script containing VBscript functions, subroutines, modules, and so forth. You can also use QuickTest to modify and debug any existing function libraries (such as **.vbs** or **.txt** files). For information on using VBScript, see "Handling VBScript Syntax Errors" on page 860 and "Understanding Basic VBScript Syntax" on page 853.

When you store a function in a function library and associate the function library with a test, the test can call the public functions in that function library. For more information, see "Working with Associated Function Libraries" on page 919. Functions that are stored in an associated function library can be accessed from the Step Generator, and the Available Keywords pane, as well as being entered manually in the Expert View.

When you store a function in a test action, it can be called only from within that action—the function cannot be called from any other action or test. This is useful if you do not want the function to be available outside of a specific action.

You can also define private functions and store them in a function library. Private functions are functions that can be called only by other functions within the same function library. This is useful if you need to reuse segments of code in your public functions.

You can define functions manually or using the Function Definition Generator, which creates the basic function definition for you automatically. Even if you prefer to define functions manually, you may still want to use the Function Definition Generator to view the syntax required to add header information, register a function to a test object, or set the function as the default method for the test object. For more information, see "Using the Function Definition Generator" on page 923.

Managing Function Libraries

You can create function libraries in QuickTest and call their functions from an action in your test. A function library is a separate QuickTest document containing VBscript functions, subroutines, modules, and so forth. Each function library opens in a separate window, enabling you to open and work on one or several function libraries at the same time. After you finish editing a function library, you can close it, leaving your QuickTest session open. You can also close all open function libraries simultaneously.

By implementing user-defined functions in function libraries and associating them with your test, you and other users can choose functions that perform complex operations, such as adding if/then statements and loops to test steps, or working with utility objects—without adding the code directly to the test. In addition, you save time and resources by implementing and using reusable functions.

QuickTest provides tools that enable you to edit and debug any function library, even if it was created using an external editor. For example, QuickTest can check the syntax of your functions, and the function library window provides the same editing features that are available in the Expert View. For more information on the options available in the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows."

Note: In QuickTest, when you open a test, QuickTest creates a local copy of the external resources that are saved to your Quality Center project. Therefore, if another user modifies an external resource saved in your Quality Center project, such as a function library, or if you modify a resource using an external editor (not QuickTest)—the changes will not be implemented in the test until the test is closed and reopened.

In contrast with this, any changes you apply to external resources saved in the file system, such as function libraries, are implemented immediately, as these files are accessed directly and are not saved as local copies when you open your test.

Creating a Function Library

You can create a new function library at any time.

To create a new function library in QuickTest:

Perform one of the following:

- Select **File** > **New** > **Function Library**
- Click the **New** button down arrow and select **Function Library**

A new function library opens.

You can now add content to your function library and/or save it. When you add content to your function library, QuickTest applies the same formatting it applies to content in the Expert View. You can modify the formatting, if needed. For more information, see "Customizing the Expert View and Function Library Windows" on page 895.

Opening a Function Library

In QuickTest, you can open any function library that is saved in the file system or your Quality Center project—even if another document is already open in QuickTest. You can only open a function library if you have read or read-write permissions for the file.

You can choose to open a function library in edit mode or read-only mode:

- **Edit mode.** Enables you to view and modify the function library. While the function library is open on your computer, other users can view the file in read-only mode, but they cannot modify it.
- **Read-only mode.** Enables you to view the function library but not modify it. By default, when you open a function library that is currently open on another computer, it opens in read-only mode. You can also choose to open a function library in read-only mode if you want to review it, but you do not want to prevent another user from modifying it.

Tip: You can also navigate directly from a function in your document to its function definition in another function library. For more information, see "Navigating to a Specific Function in a Function Library" on page 914.

To open an existing function library:

1 Perform one of the following:

- ➤ Select **File** > **Open** > **Function Library**
- ➤ Click the **Open** button down arrow and select **Function Library**

The Open Function Library dialog box opens.

Tip: To open the function library in read-only mode, select the **Open in read-only mode** check box in the Open Function Library dialog box.

---

2 In the sidebar, select the location of the file, for example, File System or Quality Center Test Resources. Browse to and select a function library, and click **Open**.

QuickTest opens the specified function library in a new window. You can now view and modify its content. For more information, see "Editing a Function Library" on page 914 and "Debugging a Function Library" on page 916.

Tips:

If the function library was recently created or opened, you can select it from the recent files list in the **File** menu.

If the function library is associated with the open test, you can also open it as follows:

- In the Resources pane, double-click the function library, or right-click the function library and select **Open Function Library**.
- In the Available Keywords panes, double-click the function library, or right-click the function library and select **Open Resource**.
- Select **Resources** > **Associated Function Libraries**. (If you select a function library that is stored in a Quality Center project, QuickTest must be connected to that project to open the associated function library.)

Saving a Function Library

After you create or edit a function library in QuickTest, you can save it to your Quality Center project or to the file system.

You can also save a function library as an attachment to a test for storage purposes only. To insert function calls from this function library into a test, you must first associate the function library with the test.

By default, QuickTest saves a function library with a **.qfl** extension, unless you specify a different extension, such as **.vbs** or **.txt**, or remove the extension altogether.

Tips:

- When you modify a function library, an asterisk (*) is displayed in the title bar until the function library is saved.
- To save all open documents, select **File** > **Save All**. QuickTest prompts you to specify a location in which to save any new files that have not yet been saved.
- To save multiple documents, select **Window** > **Windows**. In the Window dialog box, select the documents you want to save and click the **Save** button. QuickTest prompts you for the save location for any new files that have not yet been saved.
- You can also select **File** > **Save As** to save the active function library under a different name or using a different path.

To save a function library to the file system or a Quality Center project:

1. Make sure that the function library you want to save is the active document. (You can click the function library's tab to bring it into focus.)
2. Perform one of the following:

- Click the **Save** button.
- Select **File** > **Save**.
- Right-click the function library document's tab and select **Save**.

If the function library was previously saved, QuickTest saves it with your changes. Otherwise, if this is the first time you are saving this function library, the Save Function Library dialog box opens.

3. Save the function library to your Quality Center project or to the file system.

To save a function library as an attachment to a test in a Quality Center project:

1 Repeat steps 1 and 2 above, making sure that you are connected to a Quality Center project.

2 In the sidebar of the Save Function Library dialog box, click **Quality Center Test Plan**. The the title bar of the dialog box changes to Save Function Library as Attachment.

3 Browse to the test to which you want to attach the function library and double-click it. The test is listed as the last item in **Look in** path.

4 Click **Save**. The function library is saved as an attachment to the test.

Note: To insert calls to the attached function library, you need to associate it with a test. For more information, see "Associating a Function Library with a Test" on page 921.

Navigating Between Open QuickTest Documents

You can open multiple function libraries while a test is open, and you can navigate between all of your open documents.

To navigate between open QuickTest documents:

Perform one of the following:

➤ Click the tab for the required document in the Document pane.

Tip: If not all tabs are displayed due to lack of space, use the left and right scroll arrows in the Document pane to display the required document's tab.

➤ Press CTRL+TAB on your keyboard to scroll between open documents.

➤ Select the required document from the **Window** menu.

➤ Select **Window** > **Windows**, select the required document in the Windows dialog box, and click the **Activate** button.

Navigating to a Specific Function in a Function Library

After you insert a call to a function, you can navigate directly to its definition in the source document. The function definition can be located either in the same document (test or function library) or in another function library that is associated with your test. If the document containing the function definition is already open, QuickTest activates the window (brings the window into focus). If the document is closed, QuickTest opens it in read-only mode.

To navigate to a function's definition:

1. In the Expert View or function library, click in the step containing the relevant function.
2. Perform one of the following:
   - ➤ Select **Edit** > **Advanced** > **Go to Function Definition**.
   - ➤ Right-click the step and select **Go to Function Definition** from the context menu.

QuickTest activates the relevant document (if the function definition is located in another function library) and positions the cursor at the beginning of the function definition.

Editing a Function Library

You can edit a function library at any time using the QuickTest editing features that are available in the Expert View.

You can drag and drop a function (or part of it) from one document to another. (To do so, you must first separate the tabbed documents into separate document panes by clicking the **Restore Down** button (located below the QuickTest window's **Restore Down** / **Maximize** button).)

You can add steps to your function library manually or using the Step Generator. The Step Generator enables you to add steps that contain **reserved objects** (the objects that QuickTest supplies for enhancement purposes, such as utility objects), VBScript functions (such as MsgBox), utility statements (such as Wait), and user-defined functions that are defined in the same function library. IntelliSense is available for all functions defined in your action or for public functions defined in associated function libraries.

Note: In function libraries, IntelliSense does not enable you to view test object names or collections because function libraries are not connected to object repositories.

You can instruct QuickTest to check syntax by clicking the **Check Syntax** button, or by choosing **Tools** > **Check Syntax**.

Tips:

- For information on using VBScript, see "Understanding Basic VBScript Syntax" on page 853.
- To check the syntax for all function libraries associated with your test, click the **Check Syntax** button in the Resources pane of the Test Settings dialog box (**File** > **Settings** > **Resources** node). For more information, see "Defining Resource Settings for Your Test" on page 1274.

Editing a Read-Only Function Library

If you open a function library in read-only mode and then decide to modify it, you can convert the function library to an editable file—as long as the function library is not locked by another user. For more information on the options available when opening a function library, see "Opening a Function Library" on page 909.

Note: During a debug session, all documents (such as tests and function libraries) are read-only. To edit a document during a debug session, you must first stop the debug session.

To edit a read-only function library:

Select **File** > **Enable Editing** or click the **Enable Editing** button. You can now edit the function library.

Debugging a Function Library

Before you can debug a function library, you must first associate it with a test and then insert a call to at least one of its functions. You can then run the test, suspend the run session while in the context of your function library and debug the function library. For example, you can use the Debug Viewer to view, set, or modify the current value of objects or variables in your function library, or to manually run additional VBScript commands. You can step into functions (including user-defined functions), set breakpoints, stop at breakpoints, view expressions, and so forth. You can begin debugging from a specific step, or you can instruct QuickTest to pause at a specific step. For more information, see "Debugging Tests and Function Libraries" on page 1069.

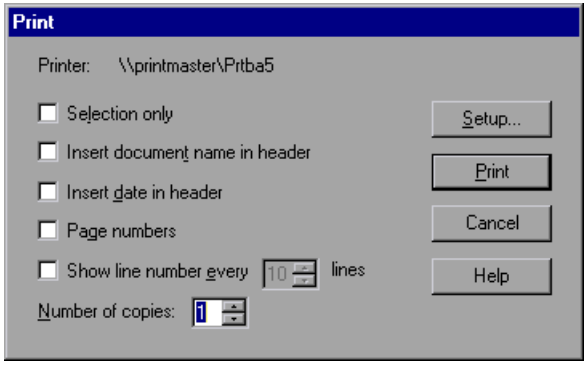
Note: During a debug session, all documents are read-only and cannot be edited. To edit a document during a debug session, you must first stop the debug session.

Printing a Function Library

You can print a function library at any time. You can also include additional information in the printout.

To print from the function library:

1 Click the **Print** button or select **File** > **Print**. The Print dialog box opens.



2 Specify the print options that you want to use:

- **Printer.** Displays the printer to which the print job will be sent. You can change the printer by clicking the **Setup** button.
- **Selection only.** Prints only the text that is currently selected (highlighted) in the function library.
- **Insert document name in header.** Includes the name of the function library at the top of the printout.
- **Insert date in header.** Includes today's date at the top of the printout. The date format is taken from your Windows regional settings.
- **Page numbers.** Includes page numbers on the bottom of the printout (for example, page 1 of 3).
- **Show line numbers every __ lines.** Displays line numbers to the left of the script lines, as specified.
- **Number of copies.** Specifies the number of times to print the document.

3 If you want to print to a different printer or change your printer preferences, click **Setup** to display the Print Setup dialog box.

4 Click **Print** to print according to your selections.

Closing a Function Library

You can close an individual function library, or if you have several function libraries open, you can close some or all of them simultaneously. If any of the function libraries are not saved, QuickTest prompts you to save them.

To close an individual function library:

Perform one of the following:

- Make sure that the function library you want to save is the active document—you can click the function library's tab to bring it into focus—and select **File** > **Close**.
- Right-click the function library document's tab and select **Close**.

- Click the **Close** button in the top right corner of the function library window.
- Select **Window** > **Windows**. In the Windows dialog box, select the function library to close if it is not already selected, and click the **Close Window(s)** button.

To close several function libraries:

Select **Window** > **Windows**. In the Windows dialog box, select the function libraries you want to close and click the **Close Window(s)** button.

To close all open function libraries:

Select **File** > **Close All Function Libraries**, or **Window** > **Close All Function Libraries**.

Working with Associated Function Libraries

In QuickTest, you can create function libraries containing functions, subroutines, modules, and so forth, and then associate the files with your test. This enables you to insert a call to a public function or subroutine in the associated function library from that test. (Public functions stored in function libraries can be called from any associated test, whereas private functions can be called only from within the same function library.)

Note: Any text file written in standard VBScript syntax can be used as a function library.

You can specify the default function libraries for all new tests in the Test Settings dialog box (**File > Settings > Resources** node). After a test is created, the list of default function libraries is integrated into the test. Therefore any changes to the default function libraries list in the Test Settings dialog box do not affect existing tests.

You can edit the list of associated function libraries for an existing test in the Resources pane or the Test Settings dialog box. For more information, see "The Resources Pane" on page 1161, and "Defining Resource Settings for Your Test" on page 1274.

Notes:

- In addition to the functions available in the associated function libraries, you can also call a function contained in any function library (or VBscript file) directly from any action using the ExecuteFile function. You can also insert ExecuteFile statements within an associated function library. For more information, see "Executing Externally-Defined Functions from Your Test" on page 948.
- You cannot debug a file that is called using an ExecuteFile statement, or any of the functions contained in the file. In addition, when debugging a test that contains an ExecuteFile statement, the execution marker may not be correctly displayed.

Working with Associated Function Libraries in Quality Center

You can associate a function library with your test, regardless of whether the function library is stored in the file system or your Quality Center project. However, if you are planning on using the function library in a business process test, you must save it in your Quality Center project.

When working with Quality Center and associated function libraries, you must save the associated function library in the Test Resources module in your Quality Center project before you specify the associated file in the Resources pane of the Test Settings dialog box. You can add a new or existing function library to your Quality Center project.

If you add an existing function library from the file system to a Quality Center project, you are actually adding a copy of that file to the project. Therefore, if you later make modifications to either of these function libraries (in the file system or in your Quality Center project), the other function library remains unaffected.

Associating a Function Library with a Test

You can associate a function library with an open test either from the Resources pane or from the currently active function library.

You can also associate function libraries with the currently open test using the associated function libraries list. For more information, see "Modifying Function Library Associations" on page 922.

To associate a function library with a test using the Resources pane:

1. In the Resources pane, right-click the **Associated Function Libraries** node in the tree and select **Associate Function Library**. The Open Function Library dialog box opens.
2. In the sidebar, select the location of the file, for example, File System or Quality Center Test Resources. Browse to and select a function library, and click **Open**.

   The function library is associated with the test and is displayed as a node under the **Associated Function Libraries** node in the tree.

To associate an open function library with a test:

1. Make sure that the test with which you want to associate the function library is open in QuickTest.
2. Create or open a function library in QuickTest. (Before continuing to the next step, make sure that the function library you want to associate with the test is the active document—you can click the function library's tab to bring it into focus.) For more information, see "Managing Function Libraries" on page 908.
3. Save the function library either in your Quality Center project or in the file system. For more information, see "Saving a Function Library" on page 911.
4. In QuickTest, select **File** > **Associate Library** '**<Function Library>**' **with** '**<Test>**', or right-click in the in the function library and select **Associate Library** '**<Function Library>**' **with** '**<Test>**'. QuickTest associates the function library with the open test.

Modifying Function Library Associations

You can modify the list of associated function libraries for a test in the Resources pane of the Test Settings dialog box, or in the Resources pane. You can add or remove function libraries from the list, and you can change their priorities.

To associate a function library with your test in the Resources pane of the Test Settings dialog box:

1 In the Test Settings dialog box (**File** > **Settings**), click the **Resources** node in the navigation bar.

2 In the **Associated function libraries** list, click the **Add** button. QuickTest displays a browse button enabling you to browse to a function library in the file system. If you are connected to a Quality Center project, QuickTest also adds [QualityCenter] to the file path, indicating that you can browse to a function library either in your Quality Center project or in the file system.

Tip: If you want to add a file from your Quality Center project but are not connected to Quality Center, press and hold the SHIFT key and click the **Add** button. QuickTest adds [QualityCenter], and you can enter the path manually. If you do, make sure there is a space after [QualityCenter]. For example: [QualityCenter] Subject\Tests

Note that QuickTest searches Quality Center project folders only when you are connected to the corresponding Quality Center project.

---

3 Select the function library you want to associate with your test and click **Open**.

To modify the priority of an associated function library:

In the list of associated function libraries in the Resources pane of the Test Settings dialog box, select the function library you want to prioritize and use the **Up** and **Down** arrows.

For more information, see “Defining Resource Settings for Your Test” on page 1274.

To remove an associated function library:

Perform one of the following:

- In the Resources pane, right-click the function library and select **Remove Function Library**, or select the function library and press the DELETE key.

- In the list of associated function libraries in the Resources pane of the Test Settings dialog box, select the function library you want to remove and click the **Remove** button.

For more information, see "Defining Resource Settings for Your Test" on page 1274.

Using the Function Definition Generator

QuickTest provides a Function Definition Generator, which enables you to generate definitions for new user-defined functions and add header information to them. You can then register these functions to a test object, if needed. You fill in the required information and the Function Definition Generator creates the basic function definition for you. After you define the function definition, you can insert the definition in your function library and associate it with your test, or you can insert the definition directly in a test script in the Expert View. Finally, you complete the function by adding its content (code).

Note: If you insert the function directly in the Expert View, the test will be able to access the function anywhere within the specific action.

If you register the function to a test object, it can be called by that test object, and is displayed in the list of available operations for that test object.

If you do not register the function to a test object, it becomes a global operation and is displayed in the list of operations in the **Operation** box in the Step Generator, and in the **Operation** column in the Keyword View, and when using IntelliSense. If you register a function, you can define it as the default operation that is displayed in the Step Generator or the Keyword View when the test object to which it is registered is selected.

Finally, you can document your user-defined function by defining the tooltip that displays when the cursor is positioned over the operation in the Step Generator, in the Keyword View, and when using IntelliSense. You can also add a sentence that describes what the step that includes the user-defined function actually does. This sentence is then displayed in the Keyword View in the **Step documentation** box of the Step Generator and in the **Documentation** column.

As you add information to the Function Definition Generator, the **Preview** area displays the emerging function definition. After you finish defining the function, you insert the definition in the active QuickTest document. If you insert it in a function library, the function will be accessible to any associated test. If you insert the function directly in a test in the Expert View, it can be called only from within the specific action. Finally, you add the content (code) of the function.

The following section provides an overview of the steps you perform when using the Function Definition Generator to create a function.

To use the Function Definition Generator:

1. Open the Function Definition Generator, as described in "Opening the Function Definition Generator" on page 925.
2. Define the function, as described in "Defining the Function Definition" on page 927.
3. Register the function to a test object, if needed, as described in "Registering a Function Using the Function Generator" on page 928.

By default, functions that are not registered to a test object are automatically defined as global functions that can be called by selecting the **Functions** category in the Step Generator, the **Operation** item in the Keyword View, or when using IntelliSense. Note that if you register the function to a test object, you can also define the function (operation) as the default operation for that selected test object.

4 Add arguments to the function, as described in "Specifying Arguments for the Function" on page 932.

5 Document the function by adding header information to it, as described in "Documenting the Function" on page 934.

6 Preview the function before finalizing it, as described in "Previewing the Function" on page 936.

7 Generate another function definition, if needed, as described in "Generating Another User-Defined Function" on page 936.

8 Finalize each function by inserting it in your active document and adding content to it, as described in "Finalizing the User-Defined Function" on page 937.

Note: Each of the steps listed in this section assumes that you have performed the previous steps.

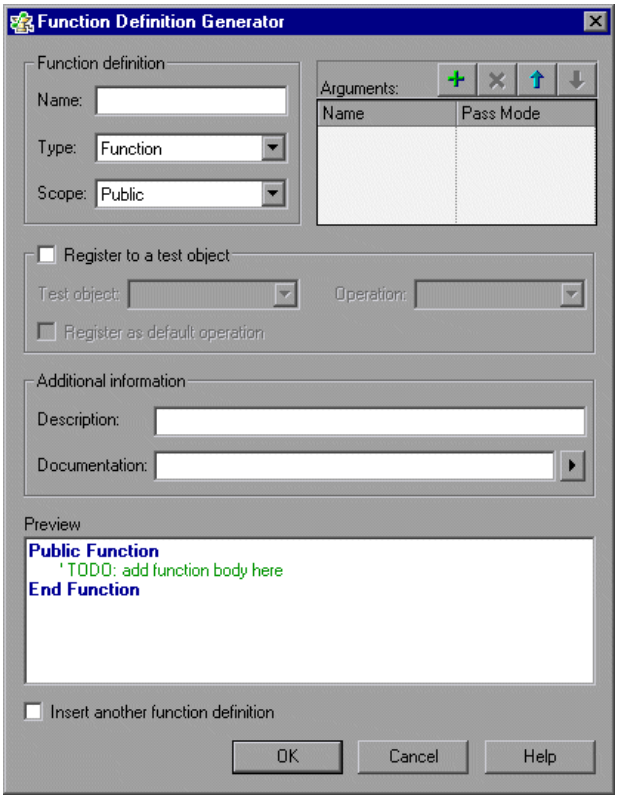
Opening the Function Definition Generator

You open the Function Definition Generator from QuickTest.

To open the Function Definition Generator:

1 Make sure that the function library or test in which you want to insert the function definition is the active document. (You can click the document's tab to bring it into focus.) This is because the Function Definition Generator inserts the function in the currently active document after you finish defining it.

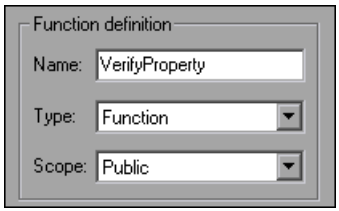
2 Select **Insert** > **Function Definition Generator** or click the **Function Definition Generator** button. The Function Definition Generator opens.



After you open the Function Definition Generator, you can begin to define a new function.

Defining the Function Definition

After you open the Function Definition Generator, you can begin defining a function.



For example, if you want to define a function that verifies the value of a specified property, you might name it VerifyProperty and define it as a public function so that it can be called from any associated test. (If you define it as private, the function can only be called from elsewhere in the same function library. Private functions cannot be registered to a test object.)

To define a function:

1 In the **Name** box, enter a name for the new function. The name should clearly indicate what the operation does so that it can be easily selected from the Step Generator or the Keyword View. Function names cannot contain non-English letters or characters. In addition, function names must begin with a letter and cannot contain spaces or any of the following characters:
! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ‘ : "" , / < > ?

Note: Do not give the user-defined function the same name as a built-in function (for example, GetLastError, MsgBox, or Print). Built-in functions take priority over user-defined functions, so if you call a user-defined function that has the same name as a built-in function, the built-in function is called instead. For a list of built-in functions, see the **Built-in functions** list in the Step Generator (**Insert** > **Step Generator**).

---

2 From the **Type** list, select **Function** or **Sub**, according to whether you want to define a function or a subroutine.

3 From the **Scope** list, select the scope of the function—either **Public** (to enable the function to be called by any test that is associated with this function library), or **Private** (to enable the function to be called only from elsewhere in the same function library). By default, the scope is set to **Public**. (Only public functions can be registered to a test object.)

Note: If you create a user-defined function manually and do not define the scope as **Public** or **Private**, it will be treated as a public function, by default.

After you define a public function, you can register the function. Alternatively, if you defined a private function, or if you do not want to register the function, you can continue by specifying arguments for the function. For more information, see "Specifying Arguments for the Function" on page 932.

Registering a Function Using the Function Generator

You can register a public function to a test object to enable the function (operation) to be performed on a test object. When you register a function to a test object, you can choose to override the functionality of an existing operation, or you can register the function as a new operation for the test object.

After you register a function to a test object, it is displayed as an operation in the Step Generator when that test object is selected, and in the Keyword View **Operation** list when that test object is selected from the **Item** list, as well as in IntelliSense and in the general **Operation** list in the Step Generator. When you register a function to a test object, it can only be called by that test object.

If you choose to register the function to a test object, the Function Definition Generator automatically adds the argument, **test_object**, as the first argument in the Arguments area in the top-right corner of the Function Definition Generator. The Function Definition Generator also automatically adds a RegisterUserFunc statement with the correct argument values immediately after your function definition.

When you register a function to a test object, you can optionally define it as the default operation for that test object. This instructs QuickTest to display the function in the **Operation** column, by default, when you or the Subject Matter Expert choose the associated test object in the **Item** list. It also enables you to select the function from IntelliSense. When you define a function as the default function for a test object, the value **True** is specified as the fourth argument of the RegisterUserFunc statement.

If you do not register the function to a specific test object, the function is automatically defined as a global function. Global functions can be called by selecting the **Functions** category in the Step Generator, or the **Operation** item in the Keyword View. A list of global functions can be viewed alphabetically in the **Operation** box when the **Functions** category is selected in the Step Generator, in the **Operation** list when the **Operation** item is selected from the **Item** list in the Keyword View, and when using IntelliSense.

During run-time, QuickTest first searches the test for the specified function and then searches the function libraries in the order in which they are listed in the Resources pane. If QuickTest finds more than one function that matches the function name in a specific test or function library, it uses the last function it finds in that test or function library. If QuickTest finds two functions with the same name in two different function libraries, it uses the function from the function library that has the higher priority. To avoid confusion, it is recommended that you verify that within the resources associated with a test, each function has a unique name.

Tip: If you choose not to register your function at this time, you can manually register it later by adding a RegisterUserFunc statement after your function as shown in the following example:

```
RegisterUserFunc "WebEdit", "MySet", "MySetFunc"
```

In this example, the MySet method (operation) is added to the WebEdit test object using the MySetFunc user-defined function. If you choose the WebEdit test object from the **Item** list in the Keyword View, the MySet operation will then be displayed in the **Operation** list (together with other registered and out of the box operations for the WebEdit test object).

You can also register your function to other test objects by duplicating (copying and pasting) the RegisterUserFunc statement and modifying the argument values as needed when you save the function code in a function library.

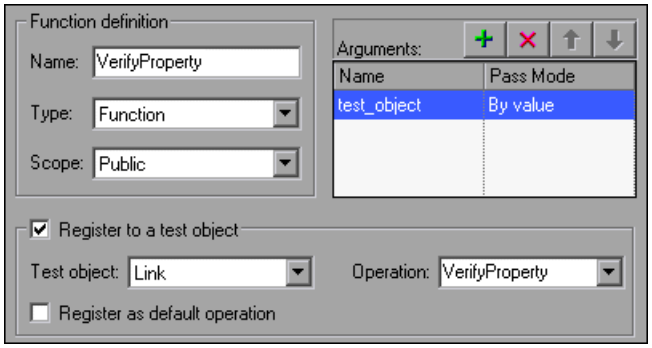
To define this function as the default function, you define the value of the fourth argument of the RegisterUserFunc statement as **True**. For example:

```
RegisterUserFunc "WebEdit", "MySet", "MySetFunc", True
```

Note: A registered or global function can only be called from a test after it is added to the test script or a function library that is associated with the test.

To register the function to a test object:

1 Select the **Register to a test object** check box. The options in this area are enabled, and a new argument, test_object, is automatically added to the list of arguments in the **Arguments** area in the top-right corner of the Function Definition Generator. (The test_object argument receives the test object to which you want to register the function.)



Note: If you clear the **Register to a test object** check box, the default test_object argument is automatically removed from the **Arguments** area (unless you renamed it).

---

2 Select a **Test object** from the list of available objects. For example, for the sample VerifyProperty function, you might want to register it to the **Link** test object.

3 Specify the **Operation** that you want to add or override for the test object.

- To define a new operation, enter a new operation name in the **Operation** box. For example, for the sample VerifyProperty function, you may want to define a new VerifyProperty operation.
- To override the standard functionality of an existing operation, select an operation from the list of available operations in the **Operation** box.

4 If you want the function to be displayed as the default operation in the **Operation** column when you or the Subject Matter Expert choose the associated item, select the **Register as default operation** check box.

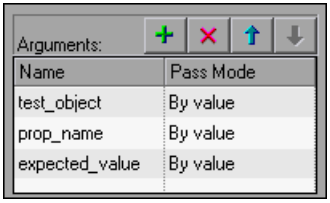
For example, if you were to define the VerifyProperty operation as the default operation for the Link test object, the value **True** would be defined as the fourth argument of the RegisterUserFunc statement, and the syntax would appear as follows:

```
RegisterUserFunc "Link", "VerifyProperty", "VerifyProperty", True
```

After you specify the test object registration information, you specify additional arguments for the function.

Specifying Arguments for the Function

After you define the basic function definition and specify the test object registration information, if any, you can specify the function's arguments.



For example, if you choose to register the function to a test object, as we did the example described in "Registering a Function Using the Function Generator" on page 928, you may want to assign the arguments prop_name (the name of the property to check) and expected_value (the expected value of the property), in addition to the first argument, test_object. You must define the required arguments for your function to run correctly.

You can list the arguments in any order. However, if you are registering the function to a test object, the first argument must always receive the test object.

To define the arguments for the function:

In the **Arguments** area, specify the arguments for the function. You can add as many arguments as needed. To ensure clarity, the name for each argument should indicate the value that needs to be entered.

- To add an argument, click  and enter a name for the argument. The argument name should clearly indicate the value that needs to be entered for the argument. Argument names may not contain non-English letters or characters. In addition, argument names must begin with a letter and cannot contain spaces or any of the following characters:
  ! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ‘ : "" , / < > ?

  For each argument, select the appropriate mode in the **Pass Mode** box to instruct QuickTest to pass the argument to the function **By value** or **By reference**.
- To remove an argument, select it and click  . The argument is removed from the Function Definition Generator.
- To set the order of the arguments, use the  and  arrows. The order of the arguments only affects the readability of the function code (except if you want to register the public function—in this case, the first argument must receive the test object).

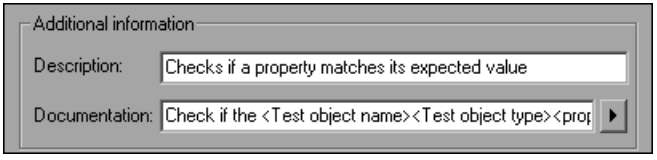
Documenting the Function

The Function Definition Generator enables you to add header information to your user-defined function. You can add a description, which is displayed as a tooltip when the cursor is positioned over the operation. You can then use this tooltip to determine which operation to choose from the list of available operations. (It is advisable to keep the description text as brief and clear as possible.)

In addition, you can add documentation that specifies exactly what a step using your function does. You can include the test object name, test object type, and any argument values in the text. You can also add text manually, as needed. This text that you add here is displayed in the Keyword View in the **Step documentation** box of the Step Generator and in the **Documentation** column. Therefore, the sentence must be clear and understandable.

For example, if you were checking a link to "HP" from a search engine, you might define the following documentation using the Function Definition Generator:

‘@Documentation Check if the <Test object name> <Test object type> <prop_name> value matches the expected value: <expected_value>.



After choosing values for the arguments in the Keyword View, the above documentation might appear as follows: Check if the "Management Software" link "text" value matches the expected value: "Business Technology Optimization (BTO) Software".

Tip: You can right-click on any column header in the Keyword View and select the **Documentation only** option to view or print a list of steps. This instructs QuickTest to display only the **Documentation** column. You can also select **Edit** > **Copy Documentation to Clipboard** and then paste the documentation in any application. Therefore, the sentence displayed for the step in this column must also be clear enough to use for manual testing instructions.

To document the function:

1 In the **Description** box, enter the text to be displayed as a tooltip when the cursor is positioned over the function name in the **Operation** list in the Step Generator, in the **Operation** column in the Keyword View, and in IntelliSense.

For example, for the sample VerifyProperty function, you may want to enter: Checks whether a property value matches the actual value.

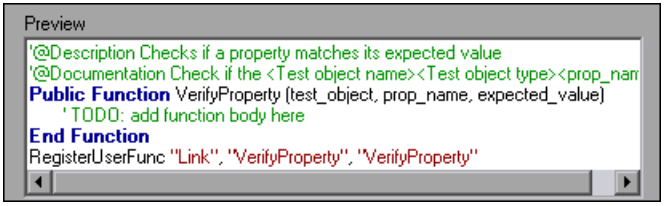
2 In the **Documentation** box, enter the text to be displayed in the **Step documentation** box in the Step Generator in the Keyword View and in the **Documentation** column of the Keyword View. You can use arguments in the **Documentation** text by clicking  and selecting the relevant argument.

If you selected the **Register to a test object** check box, clicking  also enables you to add the **Test object name** and/or **Test object type** items to the **Documentation** column from the displayed list. If you use these test object and argument items in the **Documentation** text, they are replaced dynamically by the relevant test object names and types or argument values.

Previewing the Function

The **Preview** area displays the function code as you define it, in read-only format. You can review your function and make any changes, as needed, in the various areas of the Function Definition Generator.

For example, for the sample **VerifyProperty** function, the **Preview** area displays the following code.



After you review the code (before you insert it in the active document), you can choose either to generate another function definition or to finalize the code for the function you defined.

Generating Another User-Defined Function

After you preview the code—before you insert the function in the active document—you can decide whether you want to generate an additional function definition.

Note: If you do not want to define an additional function, continue to the next section.

To generate an additional user-defined function:

1. Select the **Insert another function definition** check box and click **Insert**. QuickTest inserts the function definition in the active document and clears the data from the Function Definition Generator. The Function Definition Generator remains open.
2. Define the new function beginning from “Defining the Function Definition” on page 927.

Finalizing the User-Defined Function

After you preview the code, you insert it in the active document. If you insert it in a function library, any test associated with the function library can access the function. If you insert the function directly in a test (in the Expert View), the test can contain a call to the function from anywhere within the specific action.

After you insert the code in the required location, you can finalize the function. For example, for the VerifyProperty function, the following code would be inserted in your function library or test:

```
'@Description Checks whether a property matches its expected value
'@Documentation Check whether the <Test object name> <Test object type>
<prop_name> value matches the expected value: <expected_value>.
Public Function VerifyProperty (test_object, prop_name, expected_value)
    'TODO: add function body here
End Function
RegisterUserFunc "Link", "VerifyProperty", "VerifyProperty"
```

Tip: The RegisterUserFunc statement (in the last line) registers the VerifyProperty function to the Link test object. If you want to register the function to more than one test object, you could copy this line and duplicate it for each test object, changing the argument values, as required.

To finalize the function, you add its content (replacing the TODO comment). For example, if you want the function to verify whether the expected value of a property matches the actual property value of a specific test object, you might add the following to the body of the function:

```
Dim actual_value
    ' Get the actual property value
    actual_value = obj.GetROProperty(prop_name)
    ' Compare the actual value to the expected value
    If actual_value = expected_value Then
        Reporter.ReportEvent micPass, "VerifyProperty Succeeded", "The " &
prop_name & " expected value: " & expected_value & " matches the actual
value"
        VerifyProperty = True
    Else
        Reporter.ReportEvent micFail, "VerifyProperty Failed", "The " &
prop_name & " expected value: " & expected_value & " does not match the
actual value: " & actual_value
        VerifyProperty = False
    End If
```

To finalize the user-defined function:

1 Click **OK**. QuickTest inserts the function definition in the active document and closes the Function Definition Generator.

Note: If you define a function directly in an action, the function can be called only in that action.

---

2 In your function library or test, add content to the function code, as required, by replacing the TODO line.

Tip: To display the function in the test results tree (Test Results window) after a run session, add a Reporter.ReportEvent statement to the function code (as shown in the example above).

Note that if your user-defined function uses a default test object method, this step will appear in the Test Results window after the run session. However, you can still add a Reporter.ReportEvent statement to the function code to provide additional information and to modify the test status, if required.

---

3 If you inserted the code in a function library, you must associate the function library with a test to enable access to the user-defined functions. You also need to check its syntax to ensure that tests will have access to the functions, and that you will be able to see and use the functions. For more information, see "Working with Associated Function Libraries" on page 919.

Registering User-Defined Functions as Test Object Methods

In addition to using the QuickTest Function Definition Generator to register a function, as described in "Registering a Function Using the Function Generator" on page 928, you can also use the RegisterUserFunc statement to add new methods to test objects or to change the behavior of an existing test object method during a run session.

When you register a function to a test object, you can define it as the default operation for that test object, if required. The default operation is displayed by default in the Step Generator or the **Operation** column in the Keyword View when the test object to which it is registered is selected.

You use the UnregisterUserFunc statement to disable new methods or to return existing methods to their original QuickTest behavior.

If a you do not register a function to a test object, it becomes a global function. Global functions can be called by selecting the **Functions** category in the Step Generator, the **Operation** item in the Keyword View, or when using IntelliSense.

To register a method, you first define a function in your test or in an associated function library. You then enter a RegisterUserFunc statement at the end of the function that specifies the test object class, the function to use, and the method name that calls your function. You can register a new method for a test object class, or you can use an existing method name to (temporarily) override the existing functionality of the specified method.

Your registered method applies only to the test or function library in which you register it. In addition, QuickTest clears all function registrations at the beginning of each run session.

Preparing the User-Defined Function

You can write your user-defined function directly into your test if you want to limit its use only to the local action, or you can store the function in an associated function library to make it available to many actions and tests (recommended). If the same function name exists locally within your action and within an associated function library, QuickTest uses the function defined in the action.

When you run a statement containing a registered method, it sends the test object as the first argument. For this reason, your user-defined function must have at least one argument. Your user-defined function can have any number of arguments, or it can have only the test object argument. Make sure that if the function overrides an existing method, it has the exact syntax of the method it is replacing. This means that its first argument is the test object and the rest of the arguments match all the original method arguments.

Tip: You can use the **parent** identification property to retrieve the parent of the object represented by the first argument in your function. For example:
ParentObj = obj.GetROProperty("parent")

When writing your function, you can use standard VBScript statements as well as any QuickTest reserved objects, methods, functions, and any method associated with the test object specified in the first argument of the function.

When a function calls the test object method that it is designed to override, the standard functionality of that method is used.

For example, suppose you want to report the current value of an edit box to the Test Results before you set a new value for it. You can override the standard QuickTest Set method with a function that retrieves the current value of an edit box, reports that value to the Test Results, and then sets the new value of the edit box.

The function would look something like this:

```
Function MyFuncWithParam (obj, x)
    dim y
    y = obj.GetROProperty("value")
    Reporter.ReportEvent micDone, "previous value", y
    MyFuncWithParam=obj.Set (x)
End Function
```

Note: This function defines a return value, so that each time it is called from a test, the function returns the **Set** method argument value.

Registering User-Defined Test Object Methods

You can use the RegisterUserFunc statement to instruct QuickTest to use your user-defined function as a method of a specified test object class for the duration of a test run, or until you unregister the method.

Note: If you call an external action that registers a method (and does not unregister it at the end of the action), the method registration remains in effect for the remainder of the test that called the action.

To register a user-defined function as a test object method, use the following syntax:

RegisterUserFunc *TOClass*, *MethodName*, *FunctionName*, *SetAsDefault*

Item	Description
TOClass	Any test object class. **Note:** You cannot register a method for a QuickTest reserved object (such as **DataTable**, **Environment**, **Reporter**, and so forth).
MethodName	The name of the method you want to register (and display in QuickTest, for example, in the Keyword View and IntelliSense). If you enter the name of a method already associated with the specified test object class, your user-defined function overrides the existing method. If you enter a new name, it is added to the list of methods that the object supports.
FunctionName	The name of the user-defined function that you want to call from your test. The function can be located in your test or in any associated function library.
SetAsDefault	Indicates whether the registered function is used as the default method for the test object. When you select a test object in the Keyword View or Step Generator, the default method is automatically displayed in the **Operation** column (Keyword View) or **Operation** box (Step Generator).

Tip: If the function you are registering is defined in a function library, it is recommended to include the RegisterUserFunc statement in the function library as well so that the method will be immediately available for use in any test using that function library.

For example, suppose that the Find Flights Web page contains a **Country** edit box, and by default, the box contains the value USA. The following example registers the Set method to use the MySet function to retrieve the default value of the edit box before the new value is entered.

```
Function MySet (obj, x)
    dim y
    y = obj.GetROProperty("value")
    Reporter.ReportEvent micDone, "previous value", y
    MySet=obj.Set(x)
End Function

RegisterUserFunc "WebEdit", "Set", "MySet"
Browser("MercuryTours").Page("FindFlights").WebEdit("Country").Set "Canada"
```

For more information and examples, see the *HP QuickTest Professional Object Model Reference*.

Unregistering User-Defined Test Object Methods

When you register a method using a RegisterUserFunc statement, your method becomes a recognized method of the specified test object for the remainder of the test, or until you unregister the method. If your method overrides a QuickTest method, unregistering the method resets the method to its normal behavior. Unregistering other methods removes them from the list of methods supported by the test object.

Unregistering methods is especially important when a reusable action contains registered methods that override QuickTest methods. For example, if you do not unregister a method that uses a function defined directly within a called action, then the calling test will fail if the registered method is called again in a later action, because it will not be able to find the function definition.

If the registered function was defined in a function library, then the calling test may succeed (assuming the function library is associated with the calling test). However, unexpected results may be produced as the author of the calling test may not realize that the called action contained a registered function, and therefore, may use the registered method in later actions, expecting normal QuickTest behavior.

To unregister a user-defined method, use the following syntax:

UnRegisterUserFunc *TOClass, MethodName*

Item	Description
TOClass	The test object class for which your method is registered.
MethodName	The method you want to unregister.

For example, suppose that the Find Flights Web page contains a **Country** edit box, and by default, the box contains the value USA. The following example registers the Set method to use the MySet function to retrieve the default value of the edit box before the new value is entered. After using the registered method in a WebEdit.Set statement for the **Country** edit box, the UnRegisterUserFunc statement is used to return the Set method to its standard functionality.

```
Function MySet (obj, x)
    dim y
    y = obj.GetROProperty("value")
    Reporter.ReportEvent micDone, "previous value", y
    MySet=obj.Set(x)
End Function

RegisterUserFunc "WebEdit", "Set", "MySet"
Browser("MercuryTours").Page("FindFlights").WebEdit("Country").Set "Canada"
UnRegisterUserFunc "WebEdit", "Set"
```

Additional Tips for Working with User-Defined Functions

When working with user-defined functions, consider the following tips and guidelines:

- For an in-depth view of the required syntax, you can define a function using the Function Definition Generator and experiment with the various options.
- When you register a function, it applies to an entire test object class. You cannot register a method for a specific test object.
- If you want to call a function from additional test objects, you can copy the RegisterUserFunc line, paste it immediately after another function and replace any relevant argument values.
- If the function you are registering is defined in a function library, it is recommended to include the RegisterUserFunc statement in the function library as well so that the method will be immediately available for use in any test using that function library.
- QuickTest clears all method registrations at the beginning of each run session.
- If you use a partial run or debug option, such as **Run from step** or **Debug from step**, to begin running a test from a point after method registration was performed in a test step (and not in a function library), QuickTest does not recognize the method registration because it occurred prior to the beginning of the current run session.
- To use an Option Explicit statement in a function library associated with your test, you must include it in all the function libraries associated with the test. If you include an Option Explicit statement in only some of the associated function libraries, QuickTest ignores all the Option Explicit statements in all function libraries. You can use Option Explicit statements directly in your action scripts without any restrictions.

- Each function library must have unique variables in its global scope. If you have two associated function libraries that define the same variable in the global scope using a Dim statement or define two constants with the same name, the second definition causes a syntax error. If you need to use more than one variable with the same name in the global scope, include a Dim statement only in the last function library (since function libraries are loaded in the reverse order).
- By default, steps that use user-defined functions are not displayed in the test results tree of the Test Results window after a run session. If you want the function to appear in the test results tree, you must add a Reporter.ReportEvent statement to the function code. For example, you may want to provide additional information or to modify the test status, if required.
- If you delete a function in use from an associated function library, the test step using the function will display the  icon. In subsequent run sessions for the test, an error will occur when the step using the non-existent function is reached.
- If another user modifies a function library that is referenced by a test, or if you modify the function library using an external editor (not QuickTest), the changes will take effect only after the test is reopened.
- When more than one function with the same name exists in the test script or function library, the last function will always be called. (QuickTest searches the test script for the function prior to searching the function libraries.) To avoid confusion, make sure that you verify that within the resources associated with a test, each function has a unique name.
- If you register a method within a reusable action, it is strongly recommended to unregister the method at the end of the action (and then re-register it at the beginning of the next action if necessary), so that tests calling your action will not be affected by the method registration.

➤ You can re-register the same method to use different user-defined functions without first unregistering the method. However, when you do unregister the method, it resets to its original QuickTest functionality (or is cleared completely if it was a new method), and not to the previous registration.

For example, suppose you enter the following statements:

```
RegisterUserFunc "Link", "Click", "MyClick"
RegisterUserFunc "Link", "Click", "MyClick2"
UnRegisterUserFunc "Link", "Click"
```

After running the UnRegisterUserFunc statement, the Click method stops using the functionality defined in the MyClick2 function, and returns to the original QuickTest Click functionality, and not to the functionality defined in the MyClick function.

➤ For more information on creating functions and subroutines using VBScript, you can view the VBScript documentation from the QuickTest **Help** menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).

Executing Externally-Defined Functions from Your Test

If you decide not to associate a function library (any VBScript file) with a test, but do want to be able to call its functions, subroutines, and so forth from an action in your test or from another function library, you can do so by inserting an ExecuteFile statement in your action.

When you run your test, the ExecuteFile statement executes all global code in the function library making all definitions in the file available from the global scope of the action's script.

Note: You cannot debug a file that is called using an ExecuteFile statement, or any of the functions contained in the file. In addition, when debugging a test that contains an ExecuteFile statement, the execution marker may not be correctly displayed.

Tip: If you want to include the same ExecuteFile statement in every action you create, you can add the statement to an action template. For more information, see "Creating an Action Template" on page 462.

To execute an externally-defined function:

1. Create a VBScript file using standard VBScript syntax. For more information, see the Microsoft VBScript Language Reference (**Help** > **QuickTest Professional Help** > **VBScript Reference** > **VBScript**).
2. Store the file in any folder that you can access from the computer running your test.
3. Add an ExecuteFile statement to an action in your test using the following syntax:

   ExecuteFile *FileName*

   where *FileName* is the absolute or relative path of your VBScript file.

4 Use the functions, subroutines, and so forth, from the specified VBScript file as necessary in your action.

Notes:

- The ExecuteFile statement utilizes the VBScript **ExecuteGlobal** statement. For more information, see the Microsoft VBScript Language Reference (select **Help** > **QuickTest Professional Help** > **VBScript Reference** > **VBScript**).
- When you run an **ExecuteFile** statement within an action, you can call the functions in the file only from the current action. To make the functions in a VBScript file available to your entire test, add the file name to the associated function libraries list in the Resources pane of the Test Settings dialog box. For more information, see "Working with Associated Function Libraries" on page 919.

Part VI

Running and Analyzing Tests

32

Running Tests

After you create a test, you can run it to check the behavior of your application.

This chapter includes:

About Running Tests

When you run a test, QuickTest performs the steps it contains. If you have defined test parameters, QuickTest prompts you to enter values for them. When the run session is complete, QuickTest displays a report detailing the results. For more information on viewing the results, see Chapter 33, "Viewing Run Session Results."

If your test contains a global Data Table parameter, QuickTest runs the test once for each row in the Data Table. If your test contains a Data Table parameter for the current action data sheet, QuickTest runs the action once for each row of data in that action data sheet. You can also specify whether to run the first iteration or all iterations, for the entire test or for a specific action in the test; or to run the iterations for a specified range of data sets. For more information on test iterations, see Chapter 45, "Setting Options for Individual Tests." For more information on Data Table parameters see, Chapter 15, "Working with Actions."

You can run the entire test from the beginning, or you can run part of it. You can designate certain steps as *optional*, to enable QuickTest to bypass them instead of aborting the run if these steps do not succeed. You can update your test to change the test object descriptions, expected checkpoint values, and/or the Active Screen images and values.

You can run tests on objects with dynamic descriptions. For more information, see Chapter 5, "Managing Test Objects in Object Repositories."

You can set up a batch of tests and run them sequentially, using the QuickTest Test Batch Runner. For more information, see "Running a Test Batch" on page 966.

Note for WinRunner users: You can run WinRunner tests and call functions from WinRunner-compiled modules while running a QuickTest test. For information, see Chapter 57, "Working with WinRunner."

Running Your Entire Test

QuickTest always runs a test from the first step, unless you specify otherwise. To run a test from or to a selected step or action, you can use the **Run from Step** or **Run to Step** options. These features are useful if you want to check a specific section of the test, without running the test from the beginning or to the end. For more information, see "Running Part of Your Test" on page 956.

When you start to run a test, the Run dialog box opens to enable you to specify the location for the results and to enter the values for any test parameters you have defined.

To run a test:

1 If your test is not already open, select **File** > **Open** > **Test**.

Tip: If you recently opened your test, you can also choose it from the recent files list in the **File** menu.

---

2 Click the **Run** button in the toolbar, or select **Automation** > **Run**. The Run dialog box opens.

3 In the Run dialog box, specify the results location and the input parameter values (if applicable) for the run session. For more information, see "The Run Dialog Box: Results Location Tab" on page 960, and "The Run Dialog Box: Input Parameters Tab" on page 962.

4 Click **OK**. The Run dialog box closes and the run session starts. By default, when the run session ends, the Test Results window opens. For more information on viewing the run session results, see Chapter 33, "Viewing Run Session Results."

Note: If you cleared the **View results when run session ends** check box in the Run pane of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 44, "Setting Global Testing Options."

Tip: If you want to interrupt a run session, do either of the following:

- Click the **Pause** button in the Debug toolbar or select **Debug** > **Pause**. The run pauses. To resume running a paused run session, click the **Run** button or select **Automation** > **Run**.
- Click the **Stop** button, select **Automation** > **Stop**, or press the Stop command shortcut key. (To define a Stop command shortcut key, see "Setting Run Testing Options" on page 1253.) The run session stops and the Test Results window opens.

The run session is also interrupted if you perform a file operation (for example, open a different test or create a new test).

Running Part of Your Test

You can use the **Run from Step** option to run a selected part of your test. This enables you to check a specific section of your application or to confirm that a certain part of your test runs smoothly.

Note: You can also use the **Debug** > **Run to Step** option if you want to run a test in debug mode from the start of the test to a selected step. For more information, see "Using the Run to Step and Debug from Step Commands" on page 1076.

In the Expert View, you can use the **Run from Step** option to run your test from the selected step until the end of the action. Using **Run from Step** in this mode ignores any iterations. However, if the action contains nested actions, QuickTest runs the nested actions for the defined number of iterations of the nested action.

In the Keyword View, you can use the **Run from Step** option to run your test from the selected step until the end of the test (if the selected step is not part of a reusable action, because a reusable action needs to be called from a test, in order for the test to know from where to continue). Using **Run from Step** in this mode includes all iterations. The first iteration will run from the step you selected until the end of the test; all other iterations will run from the beginning of the test.

You can use the **Run Current Action** option to run a single action in your test. Using **Run Current Action** ignores any iterations. However, if the action contains nested actions, QuickTest runs the nested actions for the defined number of iterations.

Tips:

- If you only want to run one iteration of your test, select **Run one iteration only** from the Run pane in the Test Settings dialog box.
- If you want to run your test until a specific point within the test (and not to the end of the action or test), you can insert a breakpoint. The test will then run from the selected step or action until the breakpoint. For more information on breakpoints, see "Setting Breakpoints" on page 1079.

For more information on actions, see Chapter 15, "Working with Actions."

To run an entire action, or run a test or action from a selected step:

1 Make sure your application is in a state matching the action or step you want to run.

2 Select the action or step where you want to start running the test:

- ➤ In the Test Flow pane, select the action.
- ➤ In the Keyword View, highlight a step or action row.
- ➤ In the Expert View, place your cursor in a line of VBScript.

Make sure that the step or action you choose is not dependent on previous steps, such as a retrieved value or a parameter defined in a previous step.

3 Select **Automation** > **Run from Step** or **Run Current Action**, or right-click and select **Run from Step**. The Run dialog box opens.

4 In the Run dialog box, choose where to save the run session results, and define any input parameters you want to use, as described in "The Run Dialog Box: Results Location Tab" on page 960, and "The Run Dialog Box: Input Parameters Tab" on page 962.

Note: When running part of a test within the scope of an action, you need to specify the action's parameters, not the test parameters, in the Input Parameters tab of the Run dialog box. For more information, see "Setting Action Parameters" on page 472.

---

5 Click **OK**. The Run dialog box closes and the run session starts.

By default, when the run session ends, the Test Results window opens. For more information on viewing the run session results, see Chapter 33, "Viewing Run Session Results."

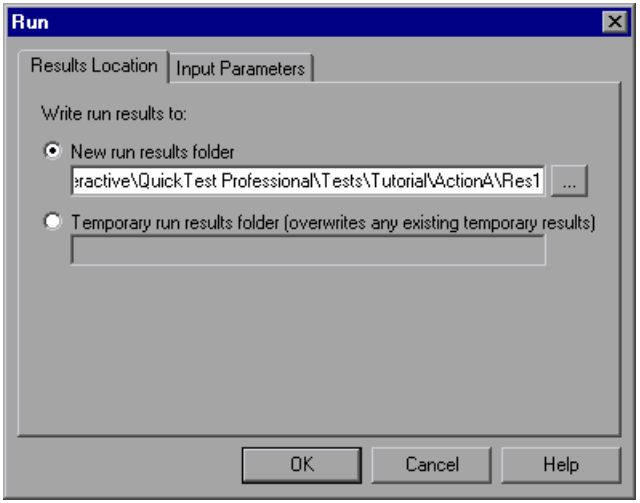
The Test Results summary displays a note indicating that the test was run using the **Run from Step** or **Run Current Action** option.

Note: If you cleared the **View results when run session ends** check box in the Run pane of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 44, "Setting Global Testing Options."

The Run Dialog Box: Results Location Tab

Description	Enables you to specify the location in which you want to save the run session results.
How to Access	The Run dialog box opens when you begin a run session in any run mode.
Learn More	**Conceptual overview**: "Running Tests" on page 953 **Primary tasks**: ➤ "Running Your Entire Test" on page 955 ➤ "Running Part of Your Test" on page 956 ➤ "Running Tests with the Maintenance Run Wizard" on page 1104 ➤ "Using the Run to Step and Debug from Step Commands" on page 1076

Below is an image of the Results Location tab in the Run dialog box:



Note: If you are running a test from a Quality Center project, the **Project name**, **Run name**, **Test set**, and **Instance** options are displayed instead of the **New run results folder** option. For more information, see "Running a Test Stored in a Quality Center Project from QuickTest" on page 1437.

Results Location Tab Options

Select one of the following options:

- **New run results folder.** This option displays the default path and folder name in which the results are saved. A new folder is created for each run. By default, the results for a QuickTest test are stored in the test folder.

  Accept the default settings, or enter a new path by typing it in the text box or clicking the browse button to locate a different folder. The folder must be new, empty, or contain only QuickTest test files.

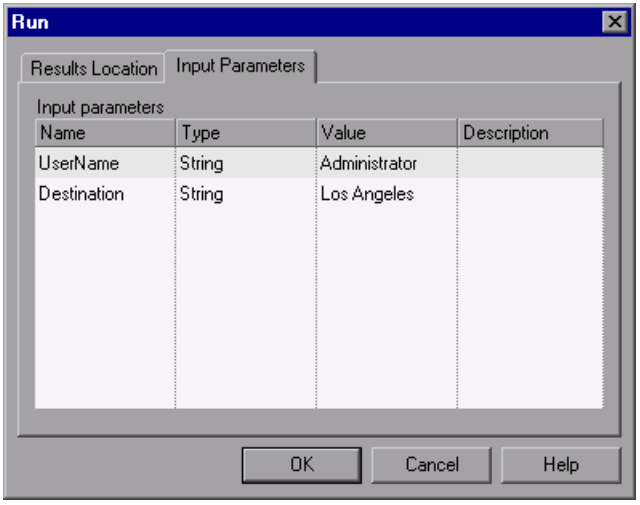
- **Temporary run results folder.** Saves the run results in a temporary folder. This option overwrites any results previously saved in this folder.

Note: QuickTest stores temporary results for all tests in **<System Drive>\Documents and Settings\<user name>\Local Settings\Temp\TempResults**. The path in the text box of the **Temporary run results folder** option cannot be changed. Additionally, if you save results to an existing results folder, the contents of the folder are deleted when the run session starts.

The Run Dialog Box: Input Parameters Tab

Description	Enables you to specify the run-time values of input parameters to be used during the run session.
How to Access	The Run dialog box opens when you begin a run session in any run mode.
Learn More	**Conceptual overview**: "Running Tests" on page 953 **Primary tasks**: ➤ "Running Your Entire Test" on page 955 ➤ "Running Part of Your Test" on page 956 ➤ "Running Tests with the Maintenance Run Wizard" on page 1104 ➤ "Using the Run to Step and Debug from Step Commands" on page 1076 **Additional related topics**: "Additional References" on page 963

Below is an image of the Input Parameters tab in the Run dialog box:



The Input Parameters tab displays the input parameters that were defined for the test (using the **File** > **Settings** > **Parameters** node).

To set the value of a parameter to be used during the run session, click in the **Value** field for the specific parameter and enter the value, or select a value from the list. If you do not enter a value, QuickTest uses the default value from the Test Settings dialog box during the run session.

Note: When running part of a test within the scope of an action (using the **Automation** > **Run from Step** or **Automation** > **Run Current Action** options), you need to specify the action's parameters, not the test parameters, in the Input Parameters tab of the Run dialog box.

Additional References

Related Concepts	➤ For more information on setting test parameters, see "Defining Parameters for Your Test" on page 1280. ➤ For more information on using parameters, see Chapter 24, "Parameterizing Values".

Using Optional Steps

An optional step is a step that is not necessarily required to successfully complete a run session. For example, suppose that when creating a test, you add login steps because the application you are testing prompts you to enter a user name and password in a login window. Suppose, too, that this particular application remembers user login details, so that you do not need to log in every time you open the application. During a run session, the application does not prompt you to enter your user name and password because it retained the information that was previously entered. In this case, the steps that you added for entering the login information are not required and should, therefore, be marked optional.

During a run session, if the object of an optional step does not exist in the application, QuickTest bypasses this step and continues to run the test. When the run session ends, a message is displayed for the step indicating that the step was not performed, but the step does not cause the run to fail.

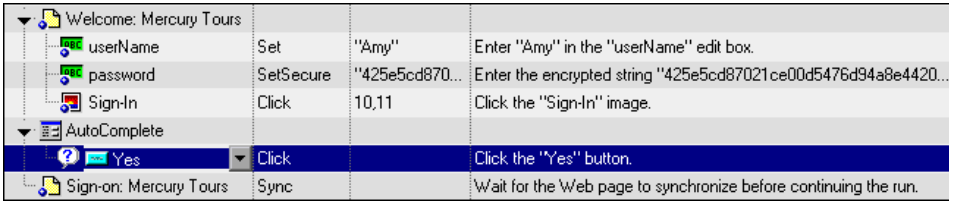
However, if, during a run session, QuickTest cannot find the object from the optional step in the object repository (for example, if the object name was modified in the test but not in the object repository, or if the object was removed from the object repository), an error message is displayed listing the required object, and the run fails.

During a recording session, QuickTest automatically marks steps that open certain dialog boxes as optional. (For a list of these dialog boxes, see "Default Optional Steps" on page 965.)

You can also manually designate steps as optional. For example, you can add conditional statements or use recovery scenarios to automatically click a button, press ENTER, or enter login information in a step. For more information, see "Using Conditional Statements" on page 797 and "Defining and Using Recovery Scenarios" on page 1329

Setting Optional Steps

To set an optional step in the Keyword View, right-click the step and select **Optional Step**. The Optional Step icon is added next to the selected step.



To add an optional step in the Expert View, add OptionalStep to the beginning of the VBScript statement. For example:

```
OptionalStep.Browser("Browser").Dialog("AutoComplete").WinButton("Yes").
Click
```

For information on working in the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows."

Default Optional Steps

By default, QuickTest considers steps that open the following dialog boxes or message boxes as optional steps:

Dialog Box / Message Box Title Bar
AutoComplete
File Download
Internet Explorer
Netscape
Enter Network Password
Error
Security Alert
Security Information
Security Warning
Username and Password Required

Running a Test Batch

You can use Test Batch Runner to run several tests in succession. The results for each test are stored in their default location.

Using Test Batch Runner, you can set up a list of tests and save the list as an **.mtb** file, so that you can easily run the same batch of tests again, at another time. You can also choose to include or exclude a test in your batch list from running during a batch run.

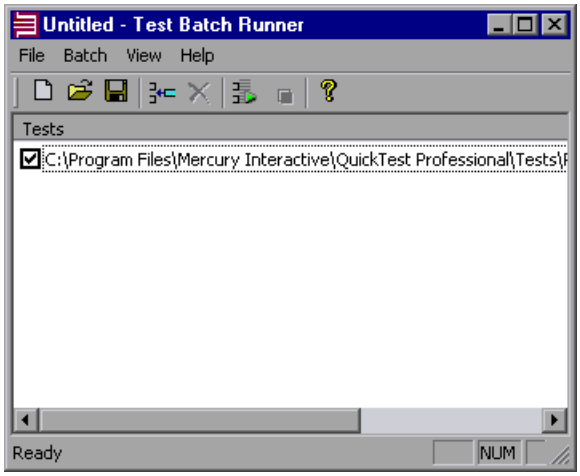
Notes:

- To enable Test Batch Runner to run tests, you must select **Allow other HP products to run tests and components** in the Run pane of the Options dialog box. For more information, see Chapter 44, "Setting Global Testing Options."
- Test Batch Runner can be used only with tests located in the file system. If you want to include tests saved in Quality Center in the batch run, you must first save the tests in the file system.
-  You can stop a test batch run at any time by clicking the **Stop** button.

To set up and run a test batch:

1. From the **Start** menu, select **Programs** > **QuickTest Professional** > **Tools** > **Test Batch Runner**. The Test Batch Runner dialog box opens.
2.  Click the **Add** button or select **Batch** > **Add**. The Open Test dialog box opens.

3 Select a test you want to include in the test batch list and click **Open**. The test is added to the list.



4 Repeat step 3 for each test you want to include in the list. By default, each test selected is added to the bottom of the list.

To insert a test at another point in the list, select the test that is to precede the test you would like to add. When you add the test, it is added above the selected test.

To remove a test from the list, select it and click the **Remove** button, or select **Batch** > **Remove**.

If you want to include a test in the list, but you do not want the test to be run during the next batch run, clear the check box next to the test name.

5 If you want to save the batch list, click the **Save** button, or select **File** > **Save**, and enter a name for the list. The file extension is **.mtb**.

6 When you are ready to run your test batch, click the **Run** button or select **Batch** > **Run**. If QuickTest is not already open, it opens and the tests run sequence begins. After the batch run is complete, you can view the results for each test in its default test results folder (**<test folder>\res#\report**).

For more information on Test Results, see Chapter 33, “Viewing Run Session Results.”

33

Viewing Run Session Results

After running a test, you can view a report of major events that occurred during the run session.

This chapter includes:

About Viewing Run Session Results

When a run session ends, you can view the run session results in the Test Results window. By default, the Test Results window opens automatically at the end of a run. If you want to change this behavior, clear the **View results when run session ends** check box in the Run pane of the Options dialog box.

The Test Results window contains a description of the steps performed during the run session. For a test that does not contain Data Table parameters, the Test Results window shows a single test iteration.

If the test contains Data Table parameters, and the test settings are configured to run multiple iterations, the Test Results window displays details for each iteration of the test run. The results are grouped by the actions in the test.

Note: You set the test to run for one or all iterations in the Run pane of the Test Settings dialog box. For more information, see "Defining Run Settings for Your Test" on page 1270.

After you run a test, the Test Results window displays all aspects of the run session and can include:

- A high-level results overview report (pass/fail status)
- The data used in all runs
- An expandable tree of the steps, specifying exactly where application failures occurred
- The exact locations in the test where failures occurred
- A still image of the state of your application at a particular step

- A movie clip of the state of your application at a particular step or of the entire test
- Detailed explanations of each step and checkpoint pass or failure, at each stage of the test
- Any system counters that were monitored for your test
- Quality Center information for your test (if the test was run from Quality Center or if a test that is stored in Quality Center is run from QuickTest and you choose to store the results in Quality Center)

Note: The Test Results window can show results with up to 300 levels in the tree hierarchy. If you have results with more than 300 nested levels, you can view the entire report by manually opening the **results.xml** file.

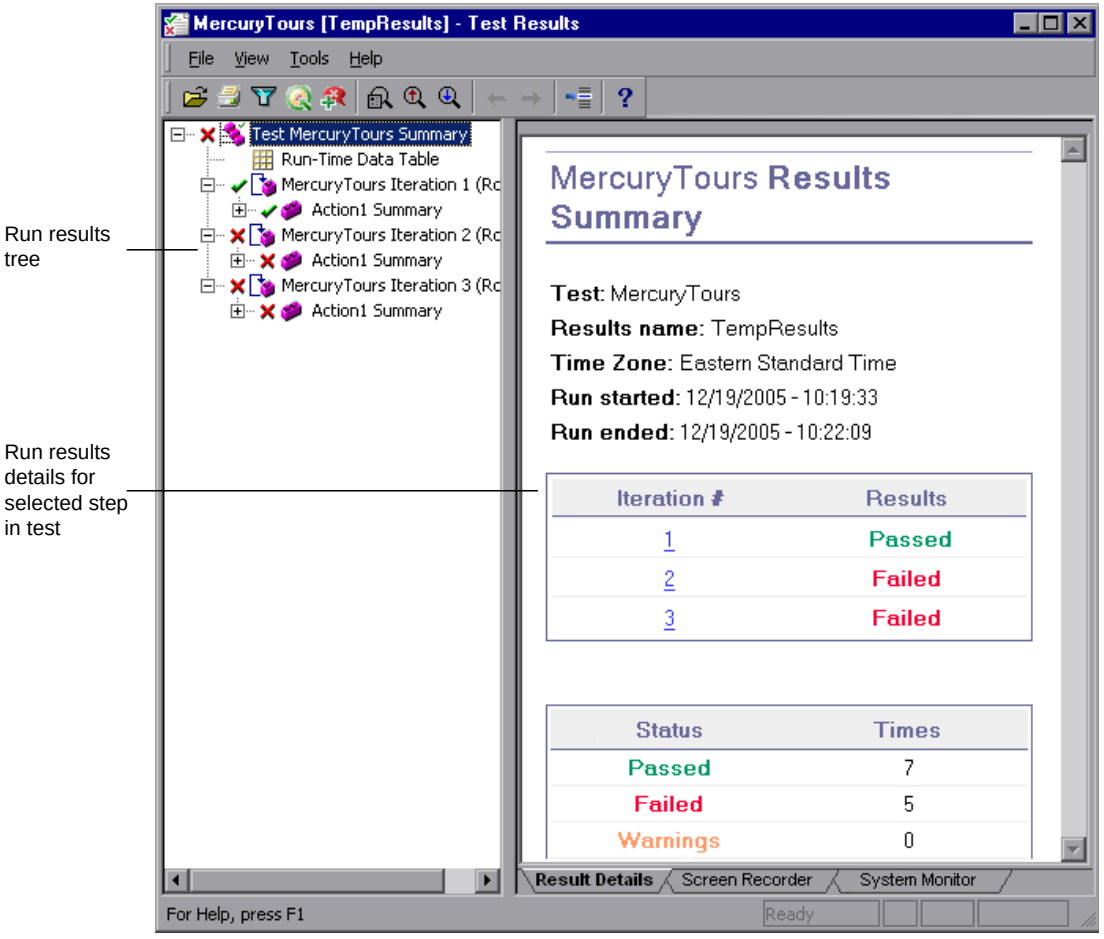
The Test Results Window

After a run session, you view the results in the Test Results window. By default, the Test Results window opens when a run session is completed. For information on changing the default setting, see "Setting Run Testing Options" on page 1253.

The left pane in the Test Results window contains the run results tree. The right pane in the Test Results window contains the details for a selected step in the run results tree. The details for a selected step may include a test summary, step details, a still image of your application, a movie of your application, or results of system counters.

You can open the Test Results window as a standalone application from the **Start** menu. To open the Test Results window, select **Start** > **Programs** > **QuickTest Professional** > **Test Results Viewer**.

The following example shows the results of a test with three iterations:



The test shown in the example above includes three iterations, as shown in the run results tree. Notice that the results for a test are organized by the test's actions.

The Test Results window contains the following key elements:

- **Test results title bar.** Displays the name of the test.
- **Menu bar.** Displays menus of available commands.
- **Run results toolbar.** Contains buttons for viewing test results (select **View > Test Results Toolbar** to display the toolbar). For more information, see "Run Results Toolbar" on page 977.
- **Run results tree.** Contains a graphic representation of the test results in the run results tree. The run results tree is located in the left pane in the Test Results window. For more information, see "Run Results Tree" on page 974.
- **Result Details tab.** Contains details of the selected node in the run results tree. The Result Details tab is located in the right pane in the Test Results window. For more information, see "Run Result Details" on page 975.
- **Screen Recorder tab.** Contains the recorded movie associated with the test results. The screen recorder tab is located in the right pane in the Test Results window. For more information, see "Viewing Still Images and Movies of Your Application" on page 992.
- **System Monitor tab.** Contains a line graph of the results for the system counters that were enabled for the test. The System Monitor tab is located in the right pane in the Test Results window. For more information see, "Viewing System Monitor Results" on page 1063.
- **Status bar.** Displays the status of the currently selected command (select **View > Status Bar** to view the status bar).

You can change the appearance of the Test Results window. For more information, see "Changing the Appearance of the Test Results Window" on page 979.

Run Results Tree

The left pane in the Test Results window displays the **run results tree**—a graphical representation of the test results:

- indicates a step that succeeded. Note that if a test does not contain checkpoints, no icon is displayed.
- indicates a step that failed. Note that this causes all parent steps (up to the root action or test) to fail as well.
- indicates a warning, meaning that the step did not succeed, but it did not cause the action or test to fail.
- indicates a step that failed unexpectedly, such as when an object is not found for a checkpoint.
- indicates an optional step that failed and therefore was ignored. Note that this does not cause the test to fail.
- indicates that the Smart Identification mechanism successfully found the object.
- indicates that a recovery scenario was activated.
- indicates that the run session was stopped before it ended.
- [password].SetSecure square brackets around a test object name indicate that the test object was created dynamically during the run session. A dynamic test object is created either using programmatic descriptions or by using an object returned by a ChildObjects method, and is not saved in the object repository.
- displays the **Run-Time Data Table**, which is a table that shows the values used to run a test containing Data Table parameters or the Data Table output values retrieved from a test while it runs.
- displays the **Maintenance Mode Update Result**, which is a table that describes the **Action** taken by Maintenance Run Wizard on a failed step and its **Details**. Displayed only for tests run in Maintenance Run Mode. For more information on Maintenance Run Mode, see Chapter 36, “Maintaining Tests.”

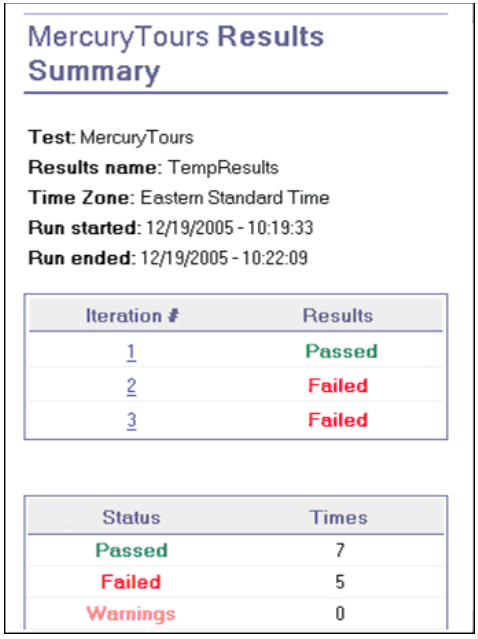
You can collapse or expand a branch in the run results tree to change the level of detail that the tree displays.

Run Result Details

By default, when the Test Results window opens, a test summary is displayed in the **Result Details** tab in the right pane in the window.

The right pane in the Test Results Window contains tabs labeled **Result Details**, **Screen Recorder**, and **System Monitor**. When you select the top node of the run results tree, the Result Details tab contains a summary of the results for your test. When you select a branch or step in the tree, the Result Details tab contains the details for that step. The Result Details tab may also include a still image of your application for the highlighted step.

When you select the top node of the run results Tree, the Result Details tab indicates the test name, results name, the start and end date and time of the run session, the number of iterations, and whether an iteration passed or failed.


MercuryTours Results Summary

Test: MercuryTours
Results name: TempResults
Time Zone: Eastern Standard Time
Run started: 12/19/2005 - 10:19:33
Run ended: 12/19/2005 - 10:22:09

Iteration #	Results
1	Passed
2	Failed
3	Failed

Status	Times
Passed	7
Failed	5
Warnings	0

The Result Details tab can also contain the following additional information:

- If an iteration contains checkpoints, the possible results are **Passed** or **Failed**. If an iteration does not contain checkpoints, the possible results are **Done** or **Failed**.
- If the Web Services Add-in is installed and was loaded during the run session, the Web Services run toolkit is displayed in the Result Details tab. The run toolkit is displayed even if the test does not include any Web Services steps.
- If the test was run in **Maintenance Run Mode**, the Result Details tab contains a **Maintenance Summary**. The **Maintenance Summary** lists the number of objects that were updated and added in your test. It also lists the number of updated and commented steps in your test. The **Object Repository Changes Report** lists the specific changes that the Maintenance Run Wizard made to the object repository and contains the following sections:
  - **Added Objects.** Lists the objects that were added to the object repository by the Maintenance Run Wizard.
  - **Object with Changed Descriptions.** Describes the changes to object properties carried out by the Maintenance Run Wizard.

  For more information on Maintenance Run Mode, see "Maintaining Tests" on page 1101.
- If the test was run from Quality Center or if a test that is stored in Quality Center is run from QuickTest and you choose to store the results in Quality Center, the name of the server, project, test set, and test instance are also shown.

Note: A test set is a group of tests selected to achieve specific testing goals. For example, you can create a test set that tests the user interface of the application or the application's performance under stress. You define test sets when working in Quality Center's test run mode. For more information, see your Quality Center documentation.

Run Results Toolbar

The Run Results toolbar contains buttons for viewing run session results.

Button	Name	Shortcut Key	Description
	**Open**	CTRL+O	Opens the Open Test Results dialog box, enabling you to open saved run results from the file system or from Quality Center. For more information, see "Opening Test Results to View a Selected Run" on page 981.
	**Print**	CTRL+P	Opens the Print dialog box, enabling you to print the results of the run session. For more information, see "Printing Test Results" on page 999.
	**Filters**	CTRL+T	Opens the Filters dialog box, enabling you to filter the information displayed. For more information, see "Filtering Test Results" on page 988.
	**Quality Center Connection**		Opens the Quality Center Connection - Server Connection dialog box, enabling you to connect to a Quality Center project. For more information, see "Connecting to and Disconnecting from Quality Center" on page 1418.
	**Add Defect**		Enables you to add a defect to your Quality Center project. If you are not currently connected to Quality Center, opens the Quality Center Connection - Server Connection dialog box. For more information, see "Submitting Defects Detected During a Run Session" on page 1013.
	**Find**	CTRL+F	Opens the Find dialog box, enabling you to search for steps with specific results, such as errors or warnings. For more information, see "Finding Results Steps" on page 990.

Button	Name	Shortcut Key	Description
	**Find Previous**	ALT+P	Finds the next step that matches the defined search filter. You define the search filter in the Find dialog box (described in "Finding Results Steps" on page 990).
	**Find Next**	ALT+N	Finds the previous step that matches the defined search filter. You define the search filter in the Find dialog box (described in "Finding Results Steps" on page 990).
	**Go to Previous Node**	BACKSPACE	Moves the cursor to the previously selected node in the run results tree. For more information, see "Navigating the Run Results Tree" on page 985.
	**Go to Next Node**	ALT+RIGHT ARROW	Moves the cursor to the node you selected in the run results tree prior to clicking the **Go to Previous Node** button. For more information, see "Navigating the Run Results Tree" on page 985.
	**Jump to Step in QuickTest**	CTRL+J	Activates the QuickTest window and highlights the step in the test corresponding to the selected node in the Test Results tree. This feature is disabled for the Action, Iteration, Run-Time Data Table, and Test Summary nodes. For more information, see "Jumping to a Step in QuickTest" on page 987.
	**Help Topics**		Opens the *HP QuickTest Professional Test Results Help*.

Changing the Appearance of the Test Results Window

By default, the Test Results window has the same look and feel as the QuickTest window, using the Microsoft Office 2003 theme. You can change the look and feel of the Test Results window, as required.

To change the appearance of the Test Results window:

In the Tests Results window, select **View > Window Theme**, and then select the way the window should appear from the list of available themes. For example, you can apply a Microsoft Office 2000 or Microsoft Windows XP theme.

Note: You can apply the Microsoft Windows XP theme to the Tests Results window only if your computer is set to use a Windows XP theme.

Tip: You can also change the theme used for the main QuickTest window. For more information, see "Changing the Appearance of the QuickTest Window" on page 27.

Viewing the Results of a Run Session

By default, the results of a run are displayed in the Test Results window at the end of the run session. (You can change the default setting in the Options dialog box. For more information on default settings for a run, see "Setting Run Testing Options" on page 1253.)

In addition, you can view the results of previous runs of the current test, and the results of other tests. You can preview test results on screen, print them or export them to an HTML file.

For more information, see:

- "Opening Test Results to View a Selected Run" on page 981
- "Navigating the Run Results Tree" on page 985
- "Viewing Result Details" on page 986
- "Jumping to a Step in QuickTest" on page 987
- "Filtering Test Results" on page 988
- "Finding Results Steps" on page 990
- "Viewing Results of Tests Run from Quality Center" on page 991
- "Viewing Still Images and Movies of Your Application" on page 992
- "Previewing Test Results" on page 997
- "Printing Test Results" on page 999
- "Exporting Test Results" on page 1001

Opening Test Results to View a Selected Run

You can view the saved results of the current test, or you can view the saved results of other tests. You can search for results in the file system or in Quality Center.

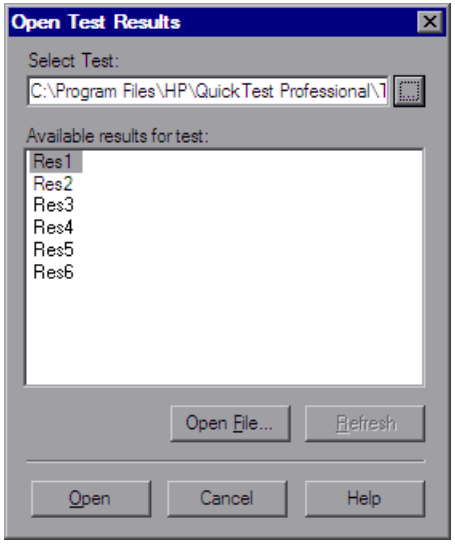
To view the saved results of the current test or other tests:

Click the **Results** button in the QuickTest window or select **Automation** > **Results**.

If there is only one saved result for the run, the run session results are displayed. If there are several results, or no results, for the current test, the Open Test Results dialog box opens.

To view the saved results of the current test or other tests from within the Test Results window:

Click the **Open** button or select **File** > **Open**. The Open Test Results dialog box opens.



The results of run sessions for the current test are listed. To view one of the results, select it and click **Open**.

Tips:

- To update the results list after you change the specified test path, click **Refresh**.
- You can open the Test Results window as a standalone application from the **Start** menu. To open the Test Results window, select **Start** > **Programs** > **QuickTest Professional** > **Test Results Viewer**.

Searching for Results in the File System or in Quality Center

By default, the results of a QuickTest test that is saved in the file system are stored in the test folder. When you run your test, you can specify a different location to store the results, using the Results Location tab of the Run dialog box. Specifying your own location for the results file can make it easier for you to locate the results file in the file system. For more information, see "The Run Dialog Box: Results Location Tab" on page 960.

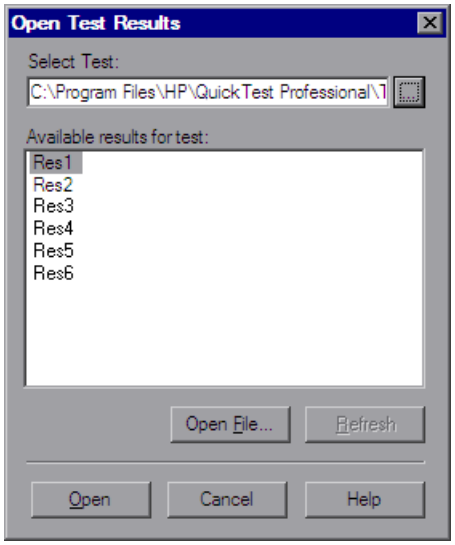
If your QuickTest test is stored in Quality Center, the results are stored in the test folder in Quality Center. You cannot change the location of the run session results.

You can search for results by test or by result file.

To search for results by test:

1  (Optional) If the test results are stored in Quality Center, in the Test Results window, select **Tools** > **Quality Center Connection** or click the **Quality Center Connection** button and connect to your Quality Center project.

2 Click the **Open** button or select **File** > **Open**. The Open Test Results dialog box opens.



3 Do one of the following:

- In the Open Test Results dialog box, enter the path of the folder that contains the results file for your test.
- Click the browse button to open the Open Test dialog box. In the sidebar, select the location of the test whose results you want to view, for example, File System or Quality Center Test Plan. Browse to and select the test, and click **Open**.

4 In the Open Test Results dialog box, highlight the test result you want to view, and click **Open**. The Test Results window displays the selected results.

For more information on working with Quality Center, see Chapter 51, “Integrating with Quality Center”.

To search for results in the file system by result file:

1 In the Open Test Results dialog box, click the **Open File** button to open the Select Results File dialog box.

2 Browse to the folder where the test results file is stored.

3 Highlight the (**.xml**) results file you want to view, and click **Open**. The Test Results window displays the selected results.

Notes:

- By default, results files for tests are stored in **<Test>\<ResultName>\Report**.
- Results files for QuickTest Professional version 6.5 and earlier are saved with a **.qtp** file extension. By default, only results files with an **.xml** extension are shown in the Select Results File dialog box. To view results files with a **.qtp** extension in the Select Results File dialog box, select **Test Results (*.qtp)** in the **Files of type** box.

Navigating the Run Results Tree

You can collapse or expand a branch in the run results tree to select the level of detail that the tree displays.

- To expand a branch, select it and click the expand (+) sign to the left of the branch icon, or press the plus key (+) on your keyboard number pad. The tree displays the details for the branch and the expand sign changes to collapse.
- When you open the Test Results window for the first time, the tree expands one level at a time. If the child branches under a parent branch were previously expanded, that state is maintained when you expand or collapse the parent branch.
- To expand a branch and all branches below it, select the branch and press the asterisk (*****) key on your keyboard number pad.
- To expand all of the branches in the run results tree, select **View** > **Expand All**; right-click a branch and select **Expand All**; or select the top level of the tree and press the asterisk (*****) key on your keyboard number pad.
- To collapse a branch, select it and click the collapse (–) sign to the left of the branch icon, or press the minus key (–) on your keyboard number pad. The details for the branch disappear in the results tree, and the collapse sign changes to expand (+).
- To collapse all of the branches in the run results tree, select **View** > **Collapse All** or right-click a branch and select **Collapse All**.

- To move between previously selected nodes within the run results tree, click the **Go to Previous Node** or **Go to Next Node** buttons.

- To find specific steps within the Test Results, click the **Find** button or select **Tools** > **Find**. For more information, see “Finding Results Steps” on page 990.

Viewing Result Details

You can view the results of an individual iteration, an action, or a step. When you select a step in the run results tree, the right side of the Test Results window contains the details of the selected step. Depending on your settings in the Run pane of the Options dialog box, the right side of the Test Results window may be split into two panes, with the bottom pane containing a still image (or in selected cases, other data) of the selected step. The right pane also includes the Screen Recorder tab, which can contain a movie from your run session, and the System Monitor tab, which can contain the results of system counters that were monitored during the test run. For more information, see "Viewing Still Images and Movies of Your Application" on page 992, "Viewing System Monitor Results" on page 1063, and "Setting Run Testing Options" on page 1253.

The results can be one of the following:

- Iterations, actions, and steps that ran successfully, but do not contain checkpoints, are marked **Done** in the right part of the Test Results window.
- Iterations, actions, and steps that contain checkpoints are marked **Passed** or **Failed** in the right part of the Test Results window and are identified by the icon ✔ or ✘ in the tree pane.
- Steps that were not successful, but did not cause the test to stop running, are marked **Warning** in the right part of the Test Results window and are identified by the icon ! or !⊗.

Note: A test, iteration, or action containing a step marked **Warning** may still be labeled **Passed** or **Done**.

Jumping to a Step in QuickTest

You can view the step in QuickTest that corresponds to a node in the run results tree.

To view the step in the test that corresponds to a node:

1 Select a node in the run results tree.

2 Perform one of the following:

a Click the **Jump to Step in QuickTest** button from the Run Results toolbar.

b Right-click and select **Jump to Step in QuickTest** from the context-sensitive menu.

c Select **View** > **Jump to Step in QuickTest**.

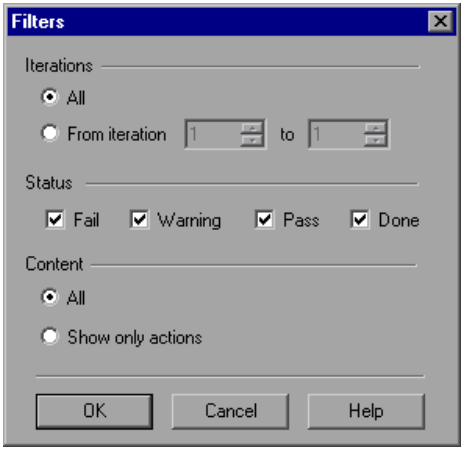
3 The QuickTest window is activated and the step is highlighted.

To jump to a step, the following conditions must be true:

- QuickTest must be running and open to the test whose results are displayed in the Test Results window.
- The test was run in a version of QuickTest that supports the Jump to Step in QuickTest functionality.
- The node has a corresponding step in QuickTest. This feature is disabled for the Action, Iteration, Run-Time Data Table, and Test Summary nodes.
- The step was not performed by a recovery scenario.
- The step was not run from the Watch or Command tabs of the Debug Viewer.
- The step is not part of an action that was run using the **LoadAndRunAction** statement. For more information, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.
- The test was saved before the run session.
- The test ran in **Normal** mode. For information on running QuickTest in **Normal** mode, see "Setting Run Testing Options" on page 1253.

Filtering Test Results

The Filters dialog box enables you to filter which iterations are displayed in the run results tree of the Test Results window.



The following options are available:

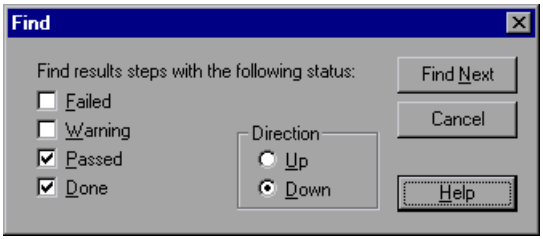
Option	Description
Iterations	(This option is available only for tests.) ➤ **All.** Displays test results from all iterations. ➤ **From iteration X to Y.** Displays the test results from the specified range of test iterations.

Option	Description
Status	➤ **Fail.** Displays the test results for the steps that failed. ➤ **Warning.** Displays the test results for the steps with the status **Warning** (steps that did not pass, but did not cause the test to fail). ➤ **Pass.** Displays the test results for the steps that passed. ➤ **Done.** Displays the test results for the steps with the status **Done** (steps that were performed successfully but did not receive a pass, fail, or warning status).
Content	(This option is available only for tests.) ➤ **All.** Displays all steps from all nodes in the test. ➤ **Show only actions.** Displays the action nodes in the test (not the specific steps in the action nodes).

Note: You can use Reporter.Filter statements in the Expert View to disable or enable the saving of selected steps, or to save only steps with **Failed** or **Warning** status. For more information on saving run session information, see "Choosing Which Steps to Report During the Run Session" on page 893 or the *HP QuickTest Professional Object Model Reference*. The Reporter.Filter statement differs from the Filters dialog box described above. The Reporter.Filter statement determines which steps are saved in the Test Results, while the Filter dialog box determines which steps are displayed at any time.

Finding Results Steps

The Find dialog box enables you to find specified steps, such as errors or warnings from within the Test Results. You can select a combination of statuses to find, for example, steps that are both **Passed** and **Done**.



The following options are available:

Option	Description
Failed	Finds a failed step in the Test Results.
Warning	Finds a step where a warning was issued.
Passed	Finds a passed step in the Test Results.
Done	Finds a step that finished its run.
Direction	Indicates whether to search up or down in the Test Results steps.

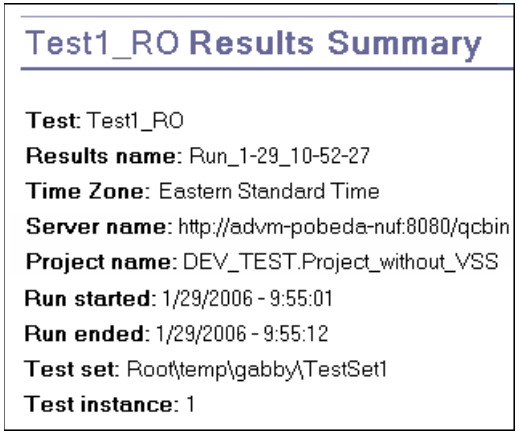
Viewing Results of Tests Run from Quality Center

When you run test sets containing QuickTest tests from Quality Center, the Quality Center server opens QuickTest on the host computer and runs the tests from that computer. All run results are then saved to the default location for those tests.

You can view the results of QuickTest test runs from Quality Center. If your results include a movie of your application, the movie can be viewed in Quality Center.

If the test was run from Quality Center or if a test that is stored in Quality Center is run from QuickTest and you choose to store the results in Quality Center, the run results contain the same information described in "The Test Results Window" on page 971, plus the following additional fields:

- **Server name.** Specifies the name of the Quality Center server from which the test was run.
- **Project name.** Specifies the Quality Center domain and project from which the test was run in the form <**domain_name.project_name**>.
- **Test set.** Specifies the location of the test set.
- **Test instance.** Specifies the instance number of the test in the test set. For example, if the same test is included twice in the test set, you can view the results of Test instance 1 and Test instance 2.



If a test that is stored in Quality Center is run from QuickTest, but you choose to store the results in a temporary location, the **Test set** and **Test instance** fields are not displayed in the results.

Viewing Still Images and Movies of Your Application

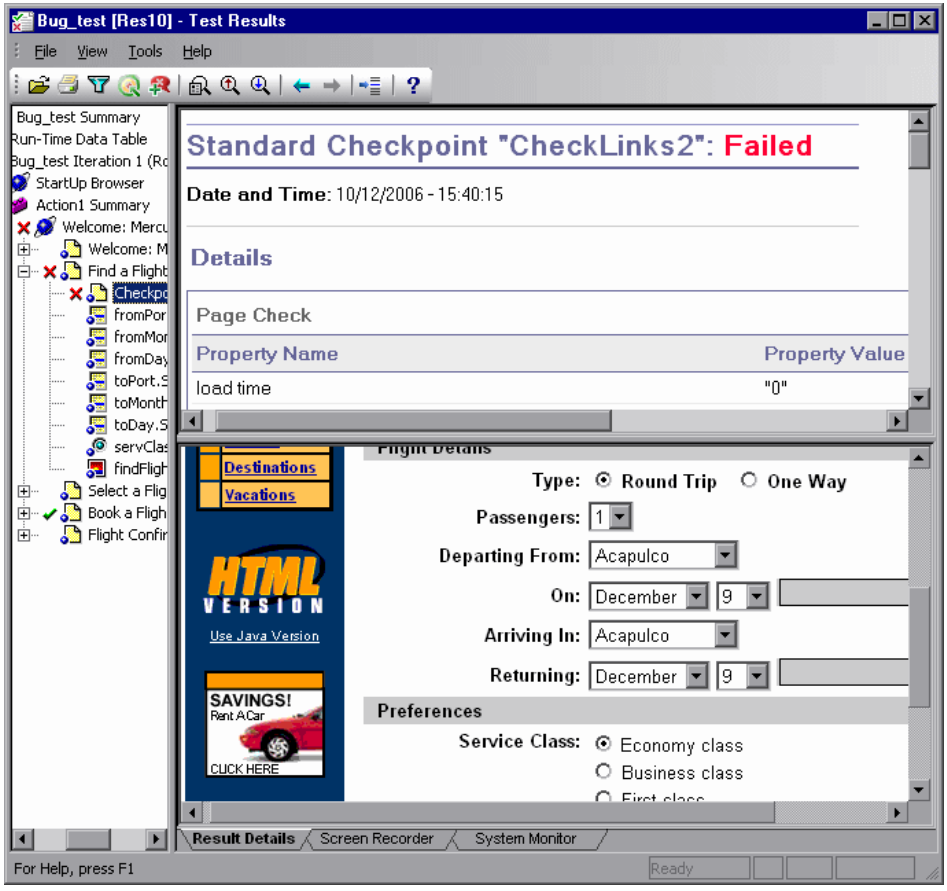
QuickTest Professional can capture still images and movies of your application during a run session. These captured files can be viewed in the Test Results window. The **Result Details** and **Screen Recorder** tabs in the right pane enable you to view either still images and text details, or a movie of your application.

Tip: You can also programmatically add an image to the **Result Details** tab using the **ReportEvent** method of the **Reporter** utility object. For more information, see the **Utility Objects** section of the *QuickTest Professional Object Model Reference*.

You configure QuickTest to capture movies of your application in the Run > Screen Capture pane of the Options dialog box. For more information, see "Setting Run Testing Options" on page 1253.

Viewing Still Images of Your Application

By default, QuickTest saves a still image of your application for failed steps. When you select a failed step in the run results tree and select the **Result Details** tab, the bottom right pane in the Test Results window displays a screen capture of your application corresponding to the highlighted step in the run results tree.



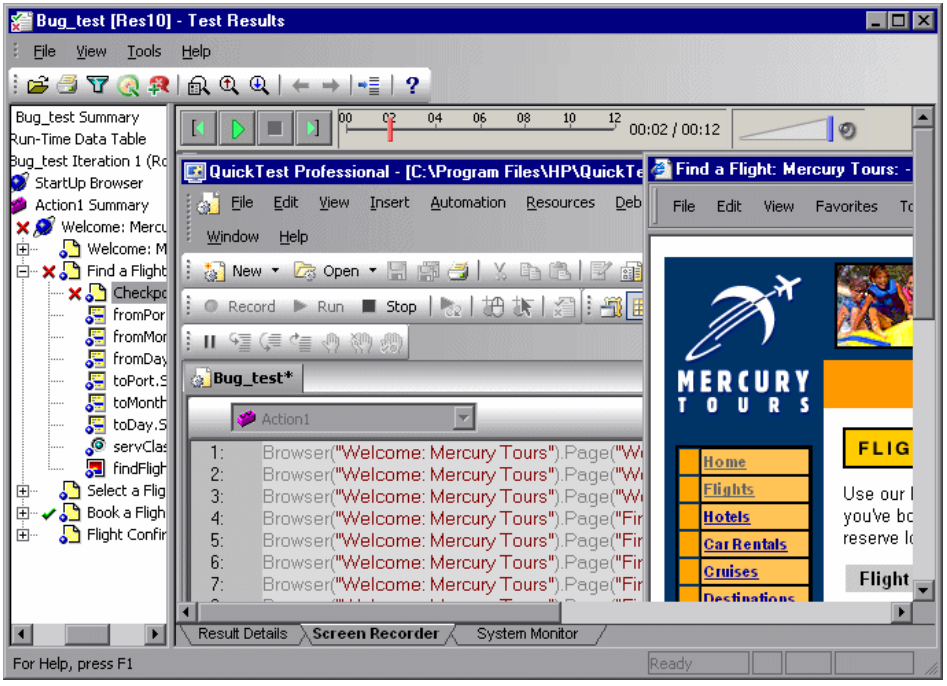
If the highlighted step does not contain an error, the right pane contains the result details with no screen capture.

You can change the conditions for when still images are saved in the test results, using the **Save still image captures to results** option in the Run > Screen Capture pane of the Options dialog box. For more information, see "The Options Dialog Box: Run > Screen Capture Pane" on page 1255.

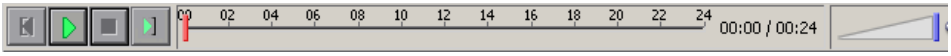
Viewing Movies of Your Run Session

QuickTest can save a movie of your application during a run session. This can be useful to help you see how your application behaved under test conditions or to debug your test. You can view the entire movie or select a particular segment to view. When you select a step in the run results tree and click the **Screen Recorder** tab, the right pane in the Test Results window displays the frame in the movie corresponding to the highlighted step in the run results tree.

You can customize the criteria QuickTest uses to save movies using the **Save movies to results** option in the Run > Screen Capture pane of the Options dialog box. For more information, see "Setting Run Testing Options" on page 1253.



The top of the Screen Recorder tab contains controls that enable you to play, pause, stop, jump to the first frame of the movie, jump to the last frame of the movie, and control the volume. You can also drag the slider bar to scroll through the movie.



Tips:

- You can double-click the right pane in the Test Results window to expand the Screen Recorder and hide the run results tree. Double-clicking again restores the Screen Recorder to its previous size and displays the run results tree. When the Screen Recorder is expanded, the playback controls at the top of the Screen Recorder automatically hide after approximately three seconds with no mouse activity, or when you click anywhere on the Screen Recorder. They reappear when you move the mouse again.
- The Screen Recorder saves a movie of your entire desktop. You can prevent the QuickTest window from partially obscuring your application while capturing the movie by minimizing QuickTest during the run session. For information on how to minimize QuickTest during run sessions, see "Customizing the QuickTest Window Layout" on page 1144.

Removing a Movie from the Test Results

You can remove a stored movie from the results of a test. This reduces the size of the test results file. To remove a movie from the test results, select **File > Remove Movie from Results**.

Exporting Captured Movie Files

You can export a captured Screen Recorder movie to a file. The file is saved as an **.fbr** file. You can view **.fbr** files in the HP Micro Recorder (as described in "Viewing Screen Recorder Movie Files in the HP Micro Player" on page 996). You can also attach **.fbr** files to defects in Quality Center. Quality Center users who have the QuickTest Add-in for Quality Center installed can view the movies from Quality Center.

To export a Screen Recorder movie:

1 Select **File > Export Movie to File**. The Save As dialog box opens, enabling you to change the default destination folder and rename the file, if required. By default, the file is named <**test name**> [<**name of run results**>], and is saved in the test results folder.

2 Click **Save** to save the exported (**.fbr**) file and close the dialog box.

Viewing Screen Recorder Movie Files in the HP Micro Player

When you capture a movie of your run session using the Screen Recorder, the movie is saved as an **.fbr** file in your test results folder. You can export **.fbr** files to any location in your file system (as described in "Exporting Captured Movie Files" on page 996). You can also view these **.fbr** files without opening the QuickTest Test Results window, using the HP Micro Player.

To play a Screen Recorder movie in the HP Micro Player:

1 Perform one of the following:

- Double-click any **.fbr** file in Windows Explorer.
- Select **Start > Programs > QuickTest Professional > Tools > HP Micro Player** and then select **File > Open** in the Micro Player to select any **.fbr** file.

The movie opens in the HP Micro Player and begins playing.

2 Use the controls at the top of the window to access a particular location in the movie or to modify the volume settings.

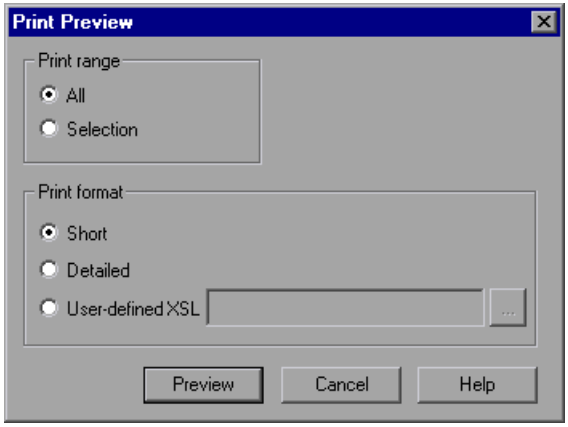
Previewing Test Results

You can preview test results on screen before you print them. You can select the type and quantity of information you want to view, and you can also display the information in a customized format.

Note: The **Print Preview** option is available only for test results created with QuickTest version 8.0 and later.

To preview the test results:

1 Select **File** > **Print Preview**. The Print Preview dialog box opens.



2 Select a **Print range** option:

- **All.** Previews the test results for the entire test.
- **Selection.** Previews test results information for the selected branch in the run results tree.

3 Select a **Print format** option:

- **Short.** Previews a summary line (when available) for each item in the run results tree. This option is only available if you selected **All** in step 2.
- **Detailed.** Previews all available information for each item in the run results tree, or for the selected branch, according to your selection in step 2. The preview includes still images associated with the steps in your run results. If a bitmap checkpoint step displays expected, actual, and difference bitmaps, these are also included.
- **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the preview, and the way it should appear. For more information, see "Customizing the Test Results Display" on page 1019.

4 Click **Preview** to preview the appearance of your test results on screen.

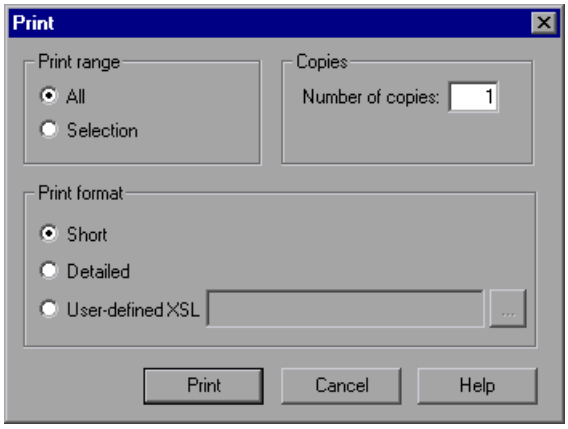
Tip: If some of the information is cut off in the preview, for example, if checkpoint names are too long to fit in the display, click the **Page Setup** button in the Print Preview window and change the page orientation from **Portrait** to **Landscape**.

Printing Test Results

You can print test results from the Test Results window. You can select the type of report you want to print, and you can also create and print a customized report.

To print the test results:

1 Click the **Print** button or select **File** > **Print**. The Print dialog box opens.



2 Select a **Print range** option:

- **All.** Prints the results for the entire test.
- **Selection.** Prints the test results for the selected branch in the run results tree.

3 Specify the **Number of copies** of the test results that you want to print.

4 Select a **Print format** option:

- **Short.** Prints a summary line (when available) for each item in the run results tree. The short report does not include still images associated with the steps in your run results. This option is only available if you selected **All** in step 2.
- **Detailed.** Prints all available information for each item in the run results tree, or for the selected branch, according to your selection in step 2. The printed report includes still images associated with the steps in your run results. If a bitmap checkpoint step displays expected, actual, and difference bitmaps, these are also included in the printed report.
- **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the printed report, and the way it should appear. For more information, see "Customizing the Test Results Display" on page 1019.

Note: The **Print format** options are available only for test results created with QuickTest version 8.0 and later.

---

5 Click **Print** to print the selected test results information to your default Windows printer.

Exporting Test Results

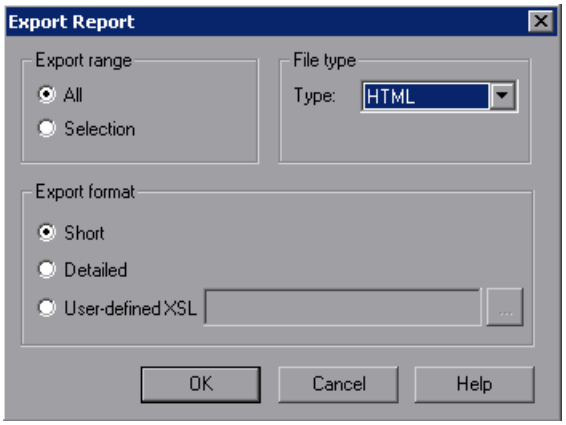
You can export the test result details to an HTML, PDF, or DOC file. This enables you to view the test results even if the QuickTest environment is unavailable. For example, you can send the file containing the test results in an e-mail to a third-party who does not have QuickTest installed. You can select the format of report you want to export, and you can also create and export a customized report. When you export test results, the information in the Result Details tab is included in the report. To export Screen Recorder or System Monitor results, use the specific export option for those tabs. For more information, see "Viewing Movies of Your Run Session" on page 994 and "Viewing System Monitor Results" on page 1063.

When selecting the file type, consider the length of time it will take to generate different document types, especially for a report with many images. HTML files generate the fastest, followed by PDF and DOC. When exporting a report with 100 or more images to a DOC file, a dialog box is displayed reminding you that it may take a long time to generate the file. The dialog box gives you the option to continue exporting with images, continue exporting without images, or to export to PDF.

When you export test results containing steps on a Web application, any screen capture images for those steps are not exported to the file. This is because for Web-based applications, the Test Results Viewer displays the HTML corresponding to the relevant Web page (with downloaded images) rather than a captured image and thus no image is saved with the report.

To export the test results:

1 Select **File** > **Export Report**. The Export Report dialog box opens.



2 Select an **Export range** option:

- **All.** Exports the results for the entire test.
- **Selection.** Exports test result information for the selected branch in the run results tree.

3 Select a **File type** from the **Type** list.

Note: To use the **DOC** format, Microsoft Word 2000 or later must be installed.

---

4 Select an **Export format** option:

- **Short.** Exports a summary line (when available) for each item in the run results tree. The short report does not include still images associated with the steps in your run results. This option is only available if you selected **All** in step 2.
- **Detailed.** Exports all available information for each item in the run results tree, or for the selected branch, according to your selection in step 2. The detailed report includes still images associated with the steps in your run results. If a bitmap checkpoint step displays expected, actual, and difference bitmaps, these are also included in the printed report.
- **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the exported report, and the way it should appear. For more information, see "Customizing the Test Results Display" on page 1019.

Note: The **Export format** options are available only for test results created with QuickTest 8.0 and later.

---

5 Click **OK**. The Save As dialog box opens. By default, the file is named <name of test> [<name of run results>], and is saved in the test results folder. You can change the default destination folder and rename the file, if required.

6 Click **Save** to save the file and close the dialog box.

Deleting Run Results

You can use the Test Results Deletion Tool to remove unwanted or obsolete test results from your system, according to specific criteria that you define. This enables you to free up valuable disk space.

You can use this tool with a Windows-style user interface, or you can use the Windows command line to run the tool in the background (silently) to directly delete results that meet criteria that you specify.

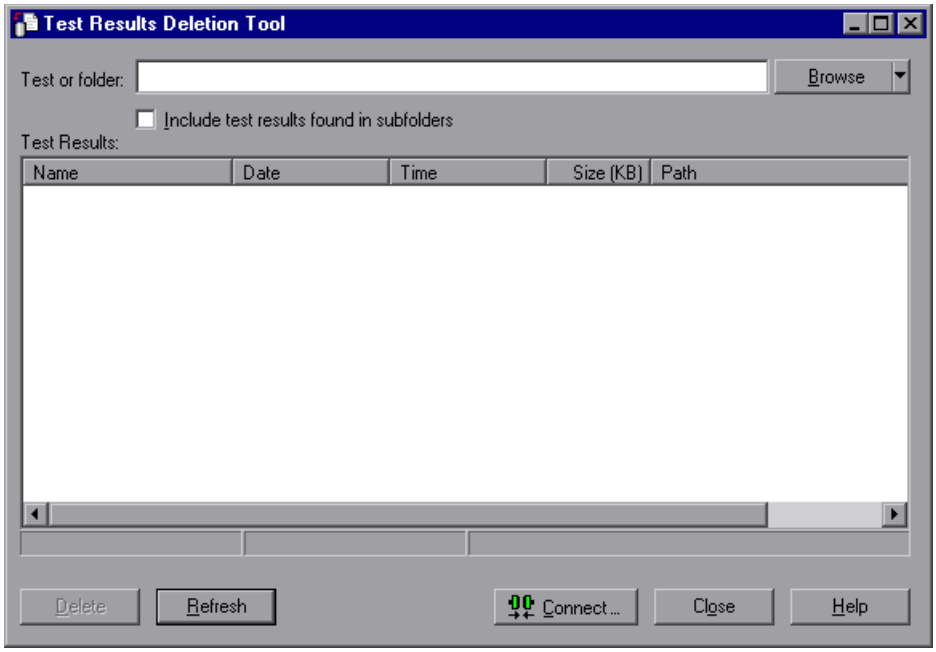
Deleting Results Using the Test Results Deletion Tool

You can use the Test Results Deletion Tool to view a list of all the test results in a specific location in your file system or in a Quality Center project. You can then delete any test results that you no longer require.

The Test Results Deletion Tool enables you to sort the test results by name, date, size, and so forth, so that you can more easily identify the results you want to delete.

To delete test results using the Test Results Deletion Tool:

1 Select **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Test Results Deletion Tool** from the **Start** menu. The Tests Results Deletion Tool window opens.



2 In the **Test or folder** box, specify the path from which you want to delete test results. When working with the file system, you can specify a test or a folder. When working with Quality Center, you cannot specify folders.

To browse to a test or folder, click the down arrow adjacent to the **Browse** button and select **Tests** or **Folders**. In the sidebar of the dialog box that opens, select the location of the test results you want to delete. Browse to and select the folder or specific test results that you want to delete, and click **Open**.

Note: To delete test results from a Quality Center database, click **Connect** to connect to Quality Center before browsing or entering the path. Specify the Quality Center test path in the standard Quality Center format. For example: [Quality Center] Subject\<folder name>\<test name>. Make sure that you have **Delete Run** permission for this Quality Center project.

For information on connecting to Quality Center, see "Connecting to and Disconnecting from Quality Center" on page 1418.

For information on Quality Center project permissions, contact your Quality Center administrator or see the section on permission settings in the *HP Quality Center Administrator Guide*.

---

3 Select **Include test results found in subfolders** if you want to view all test results contained in subfolders of the specified folder.

Note: The **Include test results found in subfolders** check box is available only for folders in the file system. It is not supported when working with tests in Quality Center.

The test results in the specified test or folder are displayed in the Test Results box, together with descriptive information for each one. You can click a column's title in the Test Results box to sort test results based on the entries in that column. To reverse the order, click the column title again.

The Delete Test Results window status bar shows information regarding the displayed test results, including the number of results selected, the total number of results in the specified location and the size of the files.

4 Select the test results you want to delete. You can select multiple test results for deletion using standard Windows selection techniques.

5 Click **Delete**. The selected test results are deleted from the system and the Quality Center database.

Tip: You can click **Refresh** at any time to update the list of test results displayed in the Test Results box.

Deleting Results Using the Windows Command Line

You can use the Windows command line to instruct the Test Results Deletion Tool to delete test results according to criteria you specify. For example, you may want to always delete test results older than a certain date or over a minimum file size.

To run the Test Results Deletion Tool from the command line:

Open a Windows command prompt and type <QuickTest installation path>\bin\TestResultsDeletionTool.exe, then type a space and type the command line options you want to use.

Note: If you use the -Silent command line option to run the Test Results Deletion Tool, all test results that meet the specified criteria are deleted. Otherwise, the Delete Test Results window opens.

Command Line Options

You can use command line options to specify the criteria for the test results that you want to delete. Following is a description of each command line option.

Note: If you add command line options that contain spaces, you must specify the option within quotes, for example:
TestResultsDeletionTool.exe -Test "F:\Tests\Keep\web objects"

-Domain *Quality_Center_domain_name*

Specifies the name of the Quality Center domain to which you want to connect. This option should be used in conjunction with the -Server, -Project, -User, and -Password options.

-FromDate *results_creation_date*

Deletes test results created after the specified date. Results created on or before this date are not deleted. The format of the date is MM/DD/YYYY.

The following example deletes all results created after November 1, 2005:

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -FromDate "11/1/2005"
```

-Log *log_file_path*

Creates a log file containing an entry for each test results file in the folder or test you specified. The log file indicates which results were deleted and the reasons why other results were not. For example, results may not be deleted if they are smaller than the minimum file size you specified.

You can specify a file path and name or use the default path and name. If you do not specify a file name, the default log file name is **TestResultsDeletionTool.log** in the folder where the Test Results Deletion Tool is located.

The following example creates a log file in **C:\temp\Log.txt**:

```
TestResultsDeletionTool.exe  -Silent -Log "C:\temp\Log.txt" -Test "C:\tests\test1"
```

The following example creates a log file named **TestResultsDeletionTool.log** in the folder where the Test Results Deletion Tool is located:

```
TestResultsDeletionTool.exe  -Silent -Log  -Test "C:\tests\test1"
```

-MinSize *minimum_file_size*

Deletes test results larger than or equal to the specified minimum file size. Specify the size in bytes.

Note: The -MinSize option is available only for test results in the file system. It is not supported when working with tests in Quality Center.

The following example deletes all results larger than or equal to 10000 bytes. Results that are smaller than 10000 bytes are not deleted:

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -MinSize "10000"
```

-Name *result_file_name*

Specifies the names of the result files to be deleted. Only results with the specified names are deleted.

You can use regular expressions to specify criteria for the result files you want to delete. For more information on regular expressions and regular expression syntax, see “Understanding and Using Regular Expressions” on page 762.

The following example deletes results with the name **Res1**:

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -Name "Res1"
```

The following example deletes all results whose name starts with **Res** plus one additional character: (For example, **Res1** and **ResD** would be deleted. **ResDD** would not be deleted.)

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -Name "Res."
```

-Password *Quality_Center_password*

Specifies the password for the Quality Center user name. This option should be used in conjunction with the -Domain, -Server, -Project, and -User options.

The following example connects to the **Default** Quality Center domain, using the server located at **http://QCServer/qcbin**, with the project named **Quality Center_Demo**, using the user name **Admin** and the password **PassAdmin**:

```
TestResultsDeletionTool.exe  -Domain "Default" -Server "http://QCServer/qcbin"
-Project "Quality Center_Demo" -User "Admin" -Password "PassAdmin"
```

-Project *Quality_Center_project_name*

Specifies the name of the Quality Center project to which you want to connect. This option should be used in conjunction with the -Domain, -Server, -User, and -Password options.

-Recursive

Deletes test results from all tests in a specified file system folder and its subfolders. When using the -Recursive option, the -Test option should contain the path of the folder that contains the tests results you want to delete (and not the path of a specific test).

The following example deletes all results in the **F:\Tests** folder and all of its subfolders:

```
TestResultsDeletionTool.exe  -Test "F:\Tests" -Recursive
```

Note: The -Recursive option is available only for folders in the file system. It is not supported when working with tests stored in Quality Center.

-Server *Quality_Center_server_path*

Specifies the full path of the Quality Center server to which you want to connect. This option should be used in conjunction with the -Domain, -Project, -User, and -Password options.

-Silent

Instructs the Test Results Deletion Tool to run in the background (silently), without the user interface.

The following example instructs the Test Results Deletion Tool to run silently and delete all results located in **C:\tests\test1**:

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1"
```

-Test *test_or_folder_path*

Sets the test or test path from which the Test Results Deletion Tool deletes test results. You can specify a test name and path, file system path, or full Quality Center path.

This option is available only when used in conjunction with the -Silent option.

Note: The -Domain, -Server, -Project, -User, and -Password options must be used to connect to Quality Center.

The following example opens the Test Results Deletion Tool with a list of the results in the **F:\Tests\Keep\webobjects** folder:

```
TestResultsDeletionTool.exe -Test "F:\Tests\Keep\webobjects"
```

The following example deletes all results in the Quality Center **Tests\webobjects** test:

```
TestResultsDeletionTool.exe -Domain "Default" -Server "http://QCServer/qcbin"
-Project "Quality Center_Demo592" -User "Admin" -Password "PassAdmin"
-Test "Subject\Tests\webobjects"
```

Tip: The -Test option can be combined with the -Recursive option to delete all test results in the specified file system folder and all its subfolders.

-UntilDate *results_creation_date*

Deletes test results created before the specified date. Results created on or after this date are not deleted. The format of the date is MM/DD/YYYY.

This option is available only when used in conjunction with the -Silent option.

The following example deletes all results created before November 1, 2005:

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -UntilDate "11/1/2005"
```

-User *Quality_Center_user_name*

Specifies the user name for the Quality Center project to which you want to connect. This option should be used in conjunction with the -Domain, -Server, -Project, and -Password options.

This option is available only when used in conjunction with the -Silent option.

Submitting Defects Detected During a Run Session

You can instruct QuickTest to automatically submit a defect to a Quality Center project for each failed step in your test. You can also manually submit a defect for a specific step to Quality Center directly from within your QuickTest Test Results window. These options are only available when you are connected to a Quality Center project.

For more information on working with Quality Center and QuickTest, see Chapter 51, "Integrating with Quality Center." For more information on Quality Center, see the *HP Quality Center User Guide*.

Manually Submitting Defects to a Quality Center Project

When viewing the results of a run session, you can submit any defects detected to a Quality Center project directly from the Test Results window.

To manually submit a defect to Quality Center:

1 Ensure that the Quality Center client is installed on your computer. (Enter the Quality Center Server URL in a browser and ensure that the Login screen is displayed.)

2 Select **Tools** > **Quality Center Connection** or click the **Quality Center Connection** button to connect to a Quality Center project. For more information on connecting to Quality Center, see Chapter 51, "Connecting to and Disconnecting from Quality Center".

Note: If you do not connect to a Quality Center project before proceeding to the next step, QuickTest prompts you to connect before continuing.

3 Select **Tools** > **Add Defect** or click the **Add Defect** button to open the New Defect dialog box in the specified Quality Center project. The New Defect dialog box opens.

4 You can modify the defect information if required. Basic information on the test and any checkpoints (if applicable) is included in the description:

5 Click **Submit** to add the defect information to the Quality Center project.

6 Click **Close** to close the Add Defect dialog box.

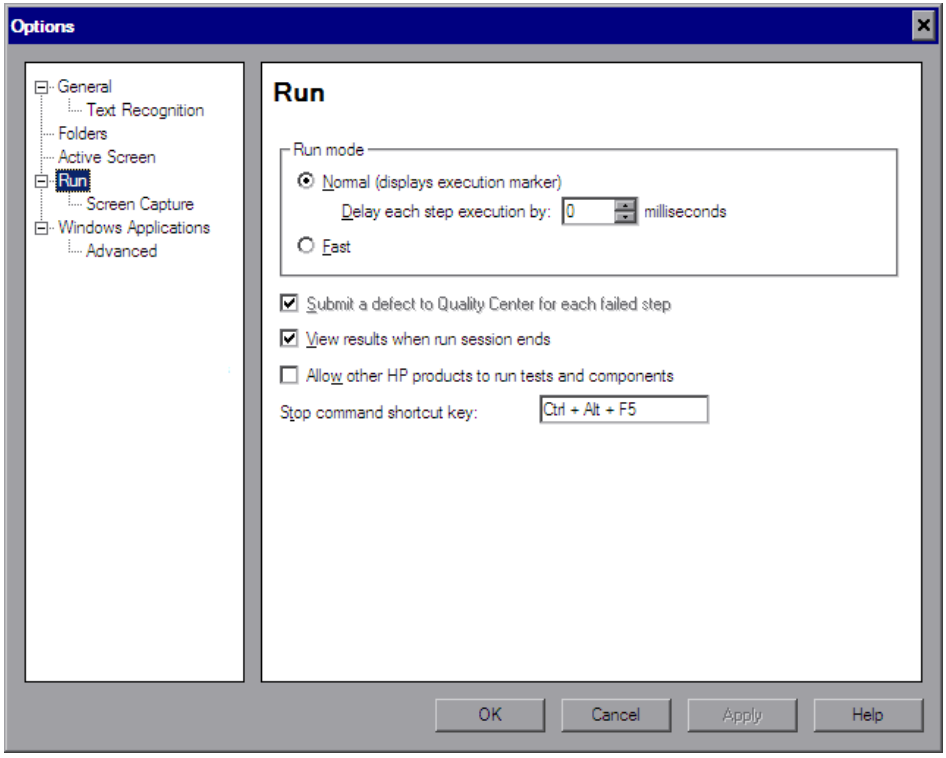
Automatically Submitting Defects to a Quality Center Project

You can instruct QuickTest to automatically submit a defect to the Quality Center project specified in the Quality Center Connection dialog box (**File > Quality Center Connection**) for each failed step in your test. You can automatically submit a defect to the Quality Center project only if the test results are stored in Quality Center.

To automatically submit defects to Quality Center:

1 Select **Tools > Options** or click the **Options** button. The Options dialog box opens.

2 Click the **Run** node.



3 Select the **Submit a defect to Quality Center for each failed step** check box.

4 Click **OK** to close the Options dialog box.

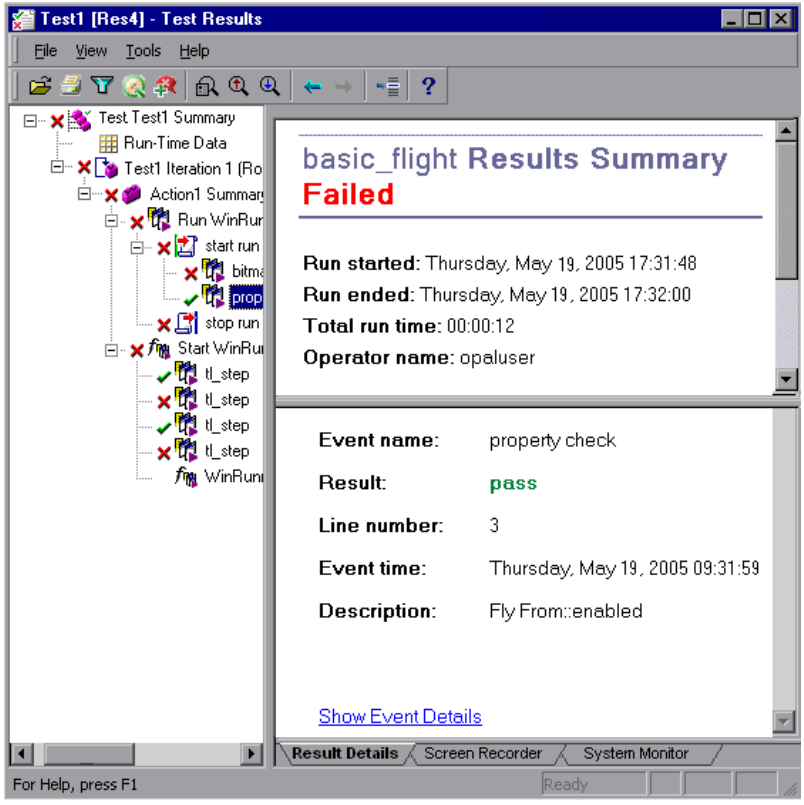
A sample of the information that is submitted to Quality Center for each defect is shown below:

```
This defect was added automatically by QuickTest Professional

Standard Checkpoint "Flight Details_4" failed

Test name: Recording
Test location: C:\Program Files\Mercury Interactive\QuickTest Professional
\Tests\Tutorial\Recording on BINDER
Action name: Action1

Operating system : Windows 2000
Host: BINDER

Additional Information:
Verification type: String Content.
Settings: Exact match - ON; Ignore space - ON; Match case - OFF.
Results: Checked  28 cells;
 Succeeded:      27;
Failed:  1
```

Viewing WinRunner Test Steps in the Test Results

If your QuickTest test includes a call to a WinRunner test, you can view detailed results of the WinRunner steps within your QuickTest Test Results window.

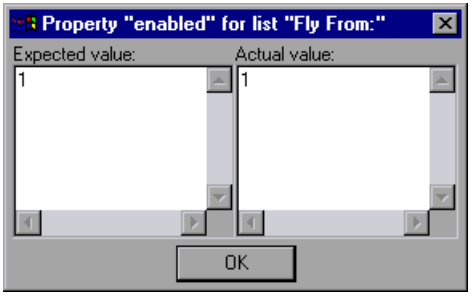
The left pane in the QuickTest test results include a node for each WinRunner event that would normally be included in the WinRunner results. When you select a node corresponding to a WinRunner test event or function call, the right pane displays a summary of the called WinRunner test or function and details about the selected event.



The start and end of the WinRunner test are indicated in the results tree by test run icons. WinRunner events are indicated by WinRunner icons. Calls to WinRunner functions are indicated by icons.

When you select a step in a WinRunner test, the top right pane displays the results summary for the WinRunner test. The summary includes the start and end time of the test, total run time, operator name, and summary results of the checkpoints performed during the test.

The bottom right pane displays the following information:

Option	Description
Event name	The name of the selected step.
Result	The status (pass or fail) of the step.
Line number	The line number of the step within the WinRunner test.
Event time	The time when the event was performed.
Description	Displays additional information on the selected step followed by a link to the WinRunner details for the step. For example, clicking the link for a GUI checkpoint that checks the enabled property of a push button displays a WinRunner dialog box similar to the following:  **Note:** You must have WinRunner installed on your computer to view WinRunner details for a selected step.

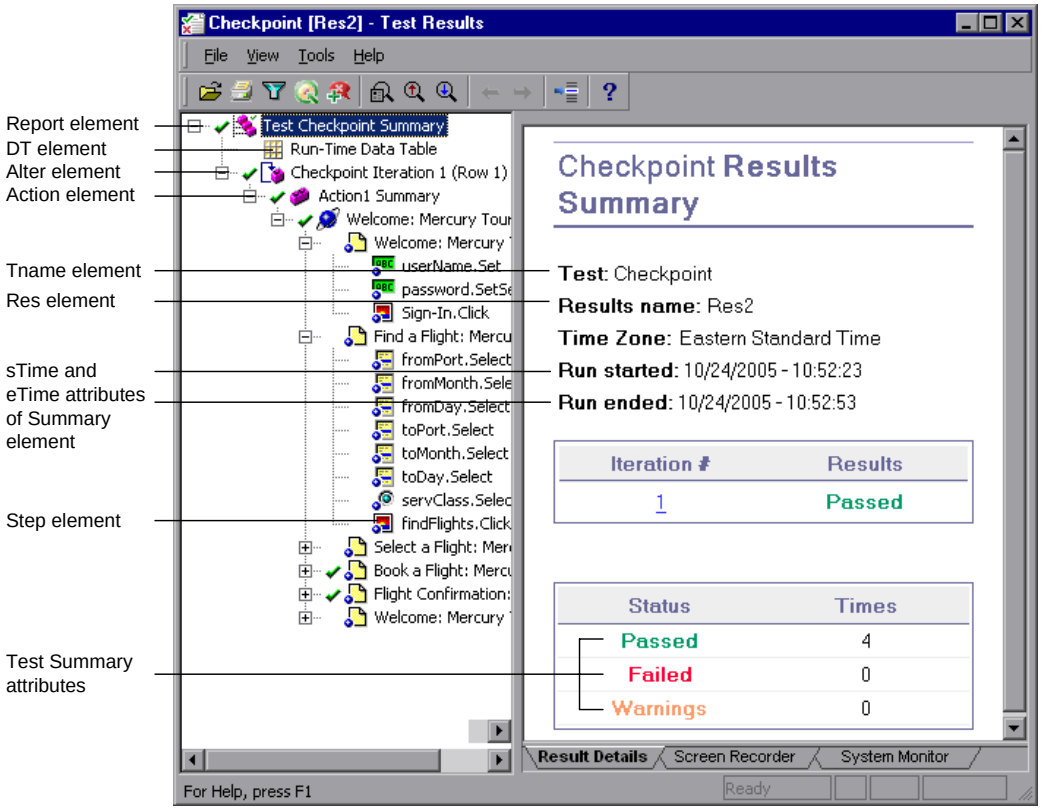
For more information on running WinRunner tests and functions from QuickTest, see Chapter 57, “Working with WinRunner.”

Customizing the Test Results Display

The results of each QuickTest run session are saved in a single **.xml** file (called **results.xml**). This **.xml** file stores information on each of the test result nodes in the display. The information in these nodes is used to dynamically create **.htm** files that are shown in the top-right pane in the Test Results window.

Each node in the run results tree is an element in the **results.xml** file. In addition, there are different elements that represent different types of information displayed in the test results. You can take test result information from the **.xml** file and use XSL to display the information you require in a customized format (either when printing from within the QuickTest Test Results window, when displaying test results in your own customized results viewer, or when exporting the test results to an HTML file).

The diagram below shows the correlation between some of the elements in the **.xml** file and the items they represent in the test results.



Tip: You can change the appearance (look and feel) of the Test Results window. For more information, see "Changing the Appearance of the Test Results Window" on page 979.

XSL provides you with the tools to describe exactly which test result information to display and exactly where and how to display, print or export it. You can also modify the **.css** file referenced by the **.xsl** file, to change the appearance of the report (for example, fonts, colors, and so forth).

For example, in the **results.xml** file, one element tag contains the name of an action, and another element tag contains information on the time at which the run session is performed. Using XSL, you could tell your customized test results viewer that the action name should be displayed in a specific place on the page and in a bold green font, and that the time information should not be displayed at all.

You may find it easier to modify the existing **.xsl** and **.css** files provided with QuickTest, instead of creating your own customized files from scratch. The files are located in **<QuickTest Installation Folder>\dat**, and are named as follows:

- **PShort.xsl.** Specifies the content of the test results report printed, or exported to an HTML file, when you select the **Short** option in the Print or Export to HTML File dialog boxes.
- **PDetails.xsl.** Specifies the content of the test results report printed, or exported to an HTML file, when you select the **Detailed** option in the Print or Export to HTML File dialog boxes.
- **PResults.css.** Specifies the appearance of the test results print preview. This file is referenced by all three **.xsl** files.

For more information on printing test results using a customized **.xsl** file, see "Printing Test Results" on page 999.

For more information on exporting the test results to a file using a customized **.xsl** file, see "Exporting Test Results" on page 1001.

For information on the structure of the XML schema, and a description of the elements and attributes you can use to customize the test results reports, see the XML Report Help (**Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Test Results Schema**).

34

Analyzing Run Session Results

You can analyze the results of a run session using the report of major events that occurred during the run session.

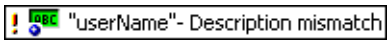
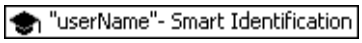
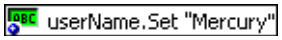
This chapter includes:

Analyzing Smart Identification Information in the Test Results

If the learned description does not enable QuickTest to identify the specified object in a step, and a Smart Identification definition is defined (and enabled) for the object, then QuickTest tries to identify the object using the Smart Identification mechanism. The following examples illustrate two possible scenarios.

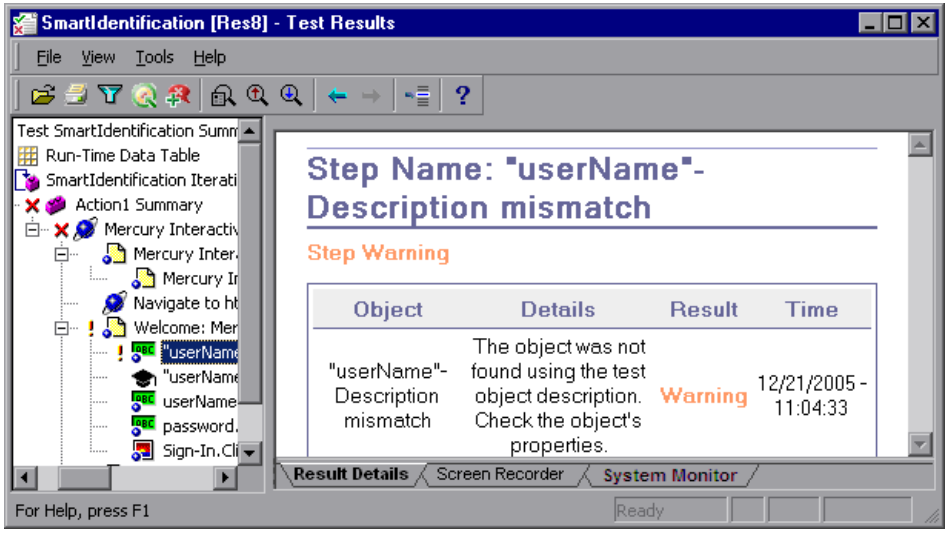
Smart Identification—No Object Matches the Learned Description

If QuickTest successfully uses Smart Identification to find an object after no object matches the learned description, the Test Results display a warning status and include the following information:

In the results tree:	In the result details:
A description mismatch icon for the missing object. For example: 	An indication that the object (for example, the userName WebEdit object) was not found.
A Smart Identification icon for the missing object. For example: 	An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to modify the learned test object description, so that QuickTest can find the object using the description in future run sessions.
The actual step performed. For example: 	Normal result details for the performed step.

For more information on the Smart Identification mechanism, see Chapter 4, “Configuring Object Identification.”

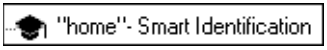
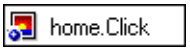
The image below shows the results for a test in which Smart Identification was used to identify the userName WebEdit object after one of the learned description property values changed.



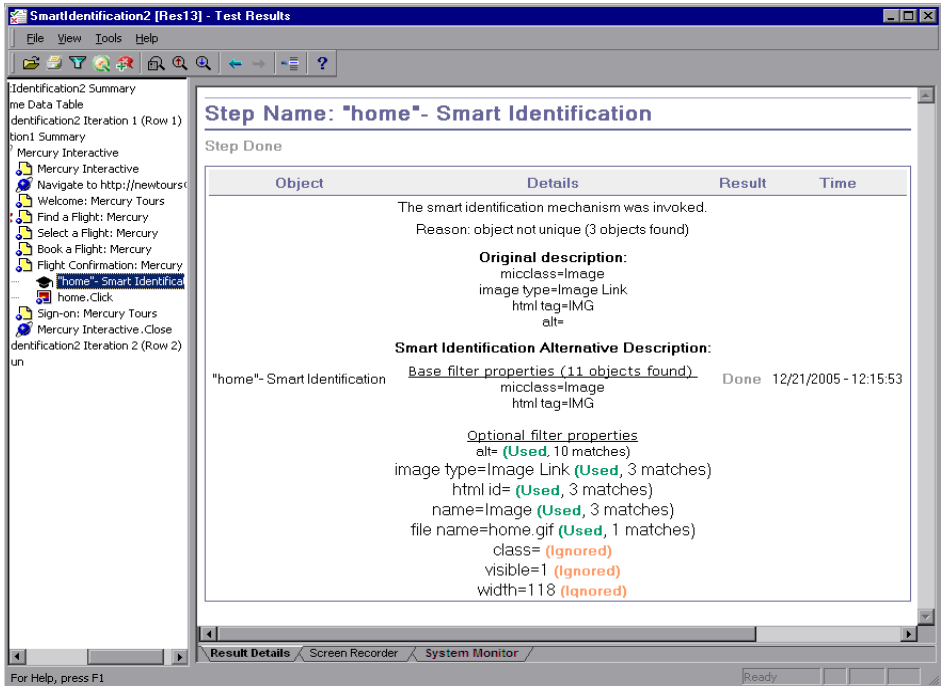
Smart Identification—Multiple Objects Match the Learned Description

If QuickTest successfully uses Smart Identification to find an object after multiple objects are found that match the learned description, QuickTest shows the Smart Identification information in the Test Results window. The step still receives a passed status, because in most cases, if Smart Identification was not used, the test object description plus the ordinal identifier could have potentially identified the object.

In such a situation, the Test Results show the following information:

In the results tree:	In the result details:
A Smart Identification icon for the missing object. For example: 	An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to create a unique object description for the object, so that QuickTest can find the object using the description in future run sessions.
The actual step performed. For example: 	Normal result details for the performed step.

The image below shows the results for a test in which Smart Identification was used to uniquely identify the Home object after the learned description resulted in multiple matches.



If the Smart Identification mechanism cannot successfully identify the object, the test fails and a normal failed step is displayed in the Test Results.

Viewing Checkpoint Results

By adding checkpoints to your test, you can compare expected values in, for example, Web pages, text strings, object properties, and tables to the values of these elements in your application. This enables you to ensure that your application functions as desired.

When you run the test, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails, which causes the test to fail. You can view the results of the checkpoint in the Test Results window.

To view the results of a checkpoint:

1 Display the test results for your test in the Test Results window. For more information, see "Viewing the Results of a Run Session" on page 980.

2 In the left pane in the Test Results window, expand the branches of the run results tree and click the branch for the checkpoint whose results you want to view. The checkpoint results are displayed in the Test Results window.

Note: By default, the bottom pane in the Result Details tab in the Test Results window displays information on the selected checkpoint only if it has the status **Failed**. You can change the conditions for when a step's image is saved, in the Run > Screen Capture pane of the Options dialog box. For more information, see "The Options Dialog Box: Run > Screen Capture Pane" on page 1255.

The information in the Test Results window and the available options are determined by the type of checkpoint you selected. For more information, see:

- "Analyzing Standard Checkpoint Results" on page 1029
- "Analyzing Table and Database Checkpoint Results" on page 1031
- "Analyzing Bitmap Checkpoint Results" on page 1033
- "Analyzing Text or Text Area Checkpoint Results" on page 1036
- "Analyzing XML Checkpoint Results" on page 1037
- "Analyzing Accessibility Checkpoint Results" on page 1048

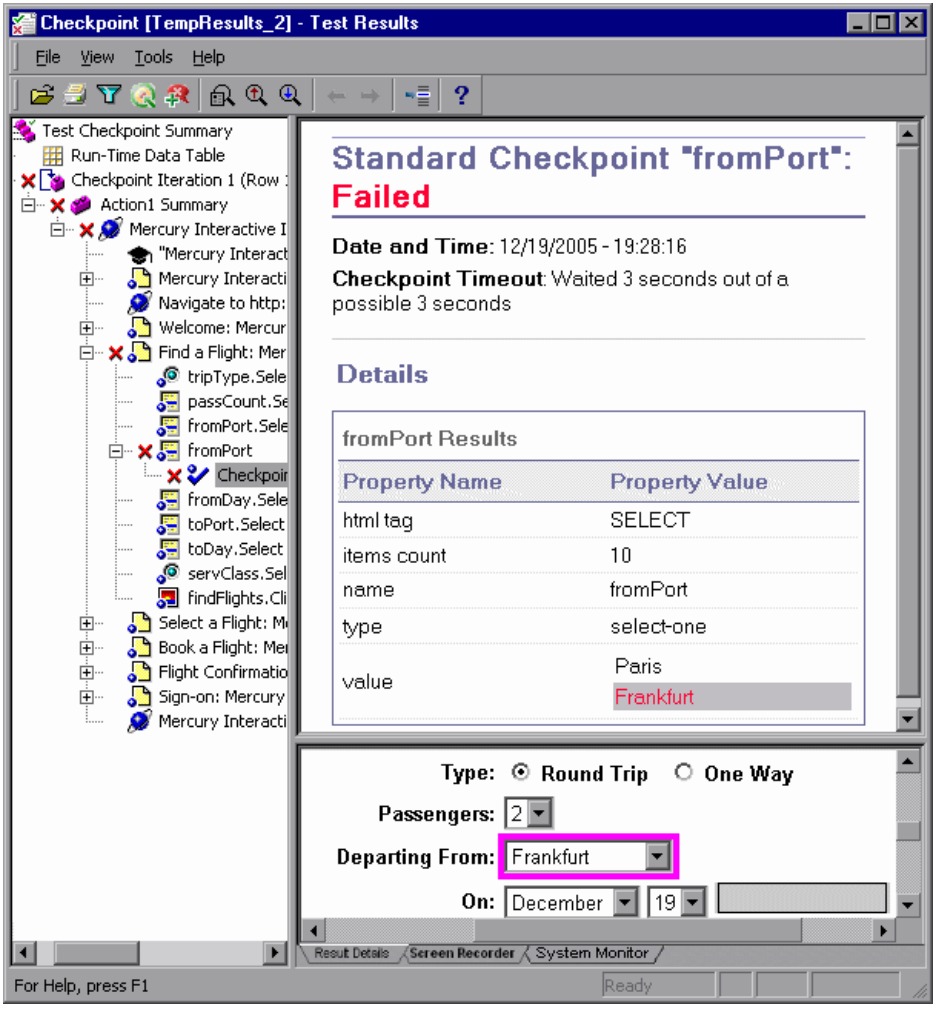
3 Select **File > Exit** to close the Test Results window.

For more information on checkpoints, see Chapter 17, "Understanding Checkpoints."

Analyzing Standard Checkpoint Results

By adding standard checkpoints to your tests, you can compare the expected values of object properties to the object's current values during a run session. If the results do not match, the checkpoint fails. For more information on standard checkpoints, see "Checking Object Property Values Using Standard Checkpoints" on page 505.

You can view detailed results of the standard checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see "Viewing Checkpoint Results" on page 1028.



The top pane in the Result Details tab displays detailed results of the selected checkpoint, including its status (**Passed** or **Failed)**, the date and time the checkpoint was run, and the portion of the checkpoint timeout interval that was used (if any). It also displays the values of the object properties that are checked, and any differences between the expected and actual property values.

The bottom pane displays the image capture for the checkpoint step (if available).

In the above example, the details of the failed checkpoint indicate that the expected results and the current results do not match. The expected value of the flight departure is **Paris**, but the actual value is **Frankfurt**.

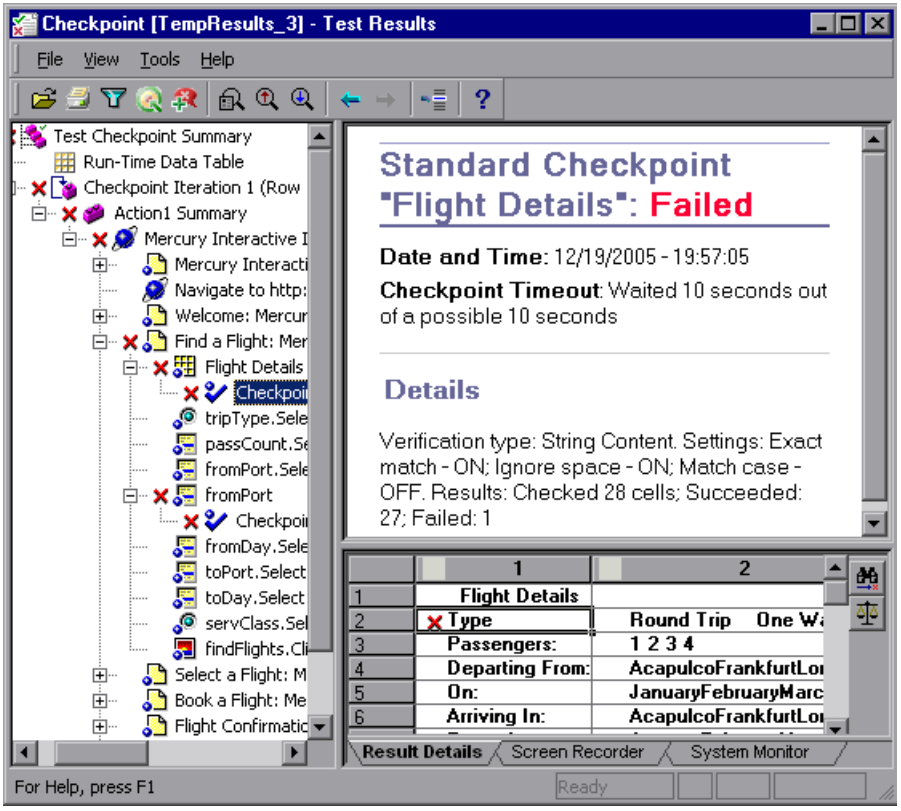
Analyzing Table and Database Checkpoint Results

By adding table checkpoints to your tests, you can check that a specified value is displayed in a cell in a table on your application. By adding database checkpoints to your tests, you can check the contents of databases accessed by your application.

The results displayed for table and database checkpoints are similar. When you run your test, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails.

For more information on table and database checkpoints, see Chapter 20, “Checking Tables” and Chapter 22, “Checking Databases.”

You can view detailed results of the table or database checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see "Viewing Checkpoint Results" on page 1028.



The top pane in the Result Details tab displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run, the verification settings you specified for the checkpoint, and the number of individual table cells or database records that passed and failed the checkpoint.

If the checkpoint failed, the bottom pane in the Result Details tab shows the table cells or database records that were checked by the checkpoint. Cell values or records that were checked are displayed in black; cell values or records that were not checked are displayed in gray. Cells or records that failed the checkpoint are marked with a failed ✘ icon.

You can click the **Next Mismatch** button in the bottom pane in the to highlight the next table cell or database record that failed the checkpoint.

You can click the **Compare Values** button in the bottom pane to display the expected and actual values of the selected table cell or database record.

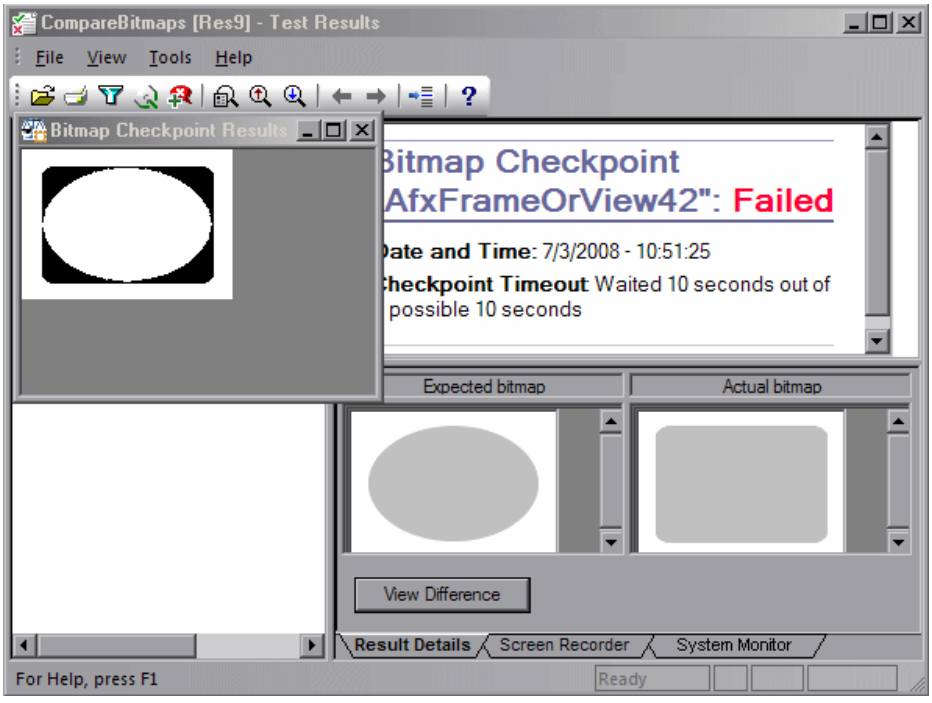
Analyzing Bitmap Checkpoint Results

By adding bitmap checkpoints to your tests, you can check the appearance of elements in your application by matching captured bitmaps. When you run your test, QuickTest compares the expected bitmap saved in the checkpoint to the actual bitmap captured from the application during the run session. If the bitmaps do not match, the checkpoint fails. For more information on bitmap checkpoints, see Chapter 19, "Checking Bitmaps."

You can view detailed results of the bitmap checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see "Viewing Checkpoint Results" on page 1028.

The top pane in the Result Details tab displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run and the portion of the checkpoint timeout interval that was used (if any).

The bottom pane in the Result Details tab shows the expected and actual bitmaps that were compared during the run session, and a **View Difference** button. When you click the **View Difference** button, QuickTest opens the Bitmap Checkpoint Results window, displaying an image that represents the difference between the expected and actual bitmaps. This image is a black-and-white bitmap that contains a black pixel for every pixel that is different in the two images.



Note: By default, the information in the bottom pane is available only if the bitmap checkpoint fails. You can change the conditions for when bitmaps are saved in the test results, using the **Save still image captures to results** option in the Run > Screen Capture pane of the Options dialog box. For more information, see "The Options Dialog Box: Run > Screen Capture Pane" on page 1255.

Considerations for Reviewing Bitmap Checkpoint Results

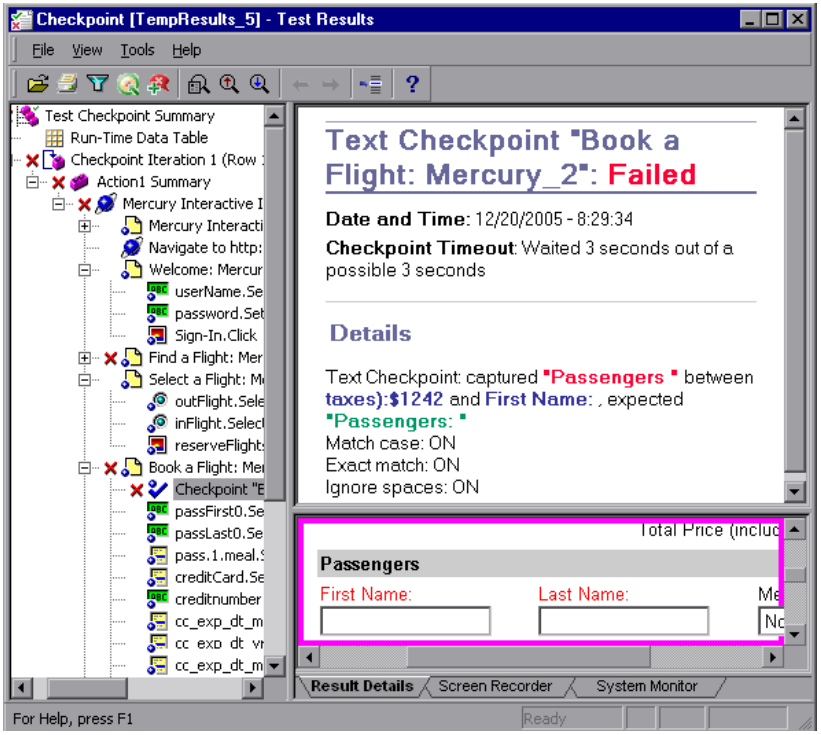
- If the checkpoint is defined to compare only a specific area of the bitmap, the test results display the actual and expected bitmaps with the selected area highlighted.
- When the dimensions of the actual and expected bitmaps are different, QuickTest fails the checkpoint without comparing the bitmaps. In this case the **View Difference** functionality is not available in the results.
- The **View Difference** functionality is not available when viewing results generated in a version of QuickTest earlier than 10.00.
- If the bitmap checkpoint is performed by a custom comparer:
  - QuickTest passes the bitmaps to the custom comparer for comparison even if their dimensions are different.
  - The top pane in the Result Details tab also displays the name of the custom comparer (as it appears in the **Comparer** box in the Bitmap Checkpoint Properties dialog box), and any additional information provided by the custom comparer.
  - The difference bitmap is provided by the custom comparer.

  For more information on using custom comparers for bitmap checkpoints, see "Fine-Tuning the Bitmap Comparison" on page 516.

Analyzing Text or Text Area Checkpoint Results

By adding text or text area checkpoints to your tests, you can check that a text string is displayed in the appropriate place in your application. When you run your test, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails. For more information on text and text area checkpoints, see Chapter 21, "Checking Text."

You can view detailed results of the text or text area checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see "Viewing Checkpoint Results" on page 1028.

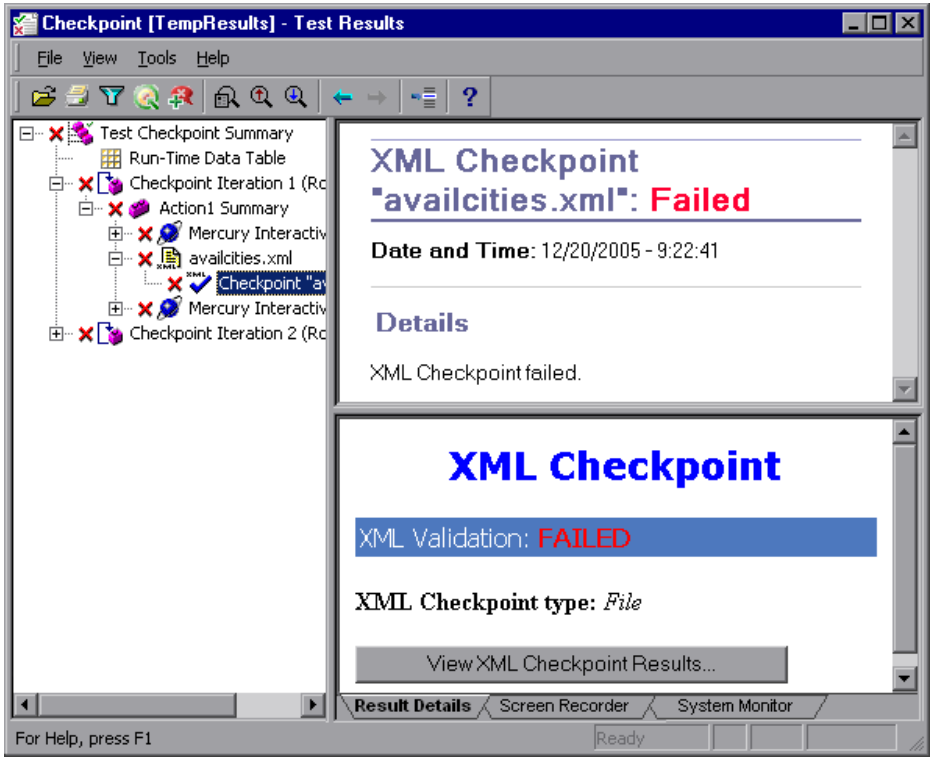


The top pane in the Result Details tab displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run and the portion of the checkpoint timeout interval that was used (if any). It also shows the expected text and actual text that was checked, and the verification settings you specified for the checkpoint.

Analyzing XML Checkpoint Results

By adding XML checkpoints to your tests, you can verify that the data and structure in your XML documents or files has not changed unexpectedly. When you run your test, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails. For more information on XML checkpoints, see Chapter 23, "Checking XML."

You can view summary results of the XML checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see "Viewing Checkpoint Results" on page 1028.



The top pane in the Result Details tab displays the checkpoint step results.

The bottom pane in the Result Details tab shows the details of the schema validation (if applicable) and a summary of the checkpoint results. If the schema validation failed, the reasons for the failure are also shown.

If the checkpoint failed, you can view details of each check performed in the checkpoint by clicking **View XML Checkpoint Results** in the bottom pane in the Result Details tab. The XML Checkpoint Results window opens, displaying details of the checkpoint's failure.

Note: By default, if the checkpoint passes, the **View XML Checkpoint Results** button is not available. The availability of these detailed results is dependent on the **Save still image captures to results** setting in the Run > Screen Capture pane of the Options dialog box. For more information, see "The Options Dialog Box: Run > Screen Capture Pane" on page 1255.

Understanding the XML Checkpoint Results Window

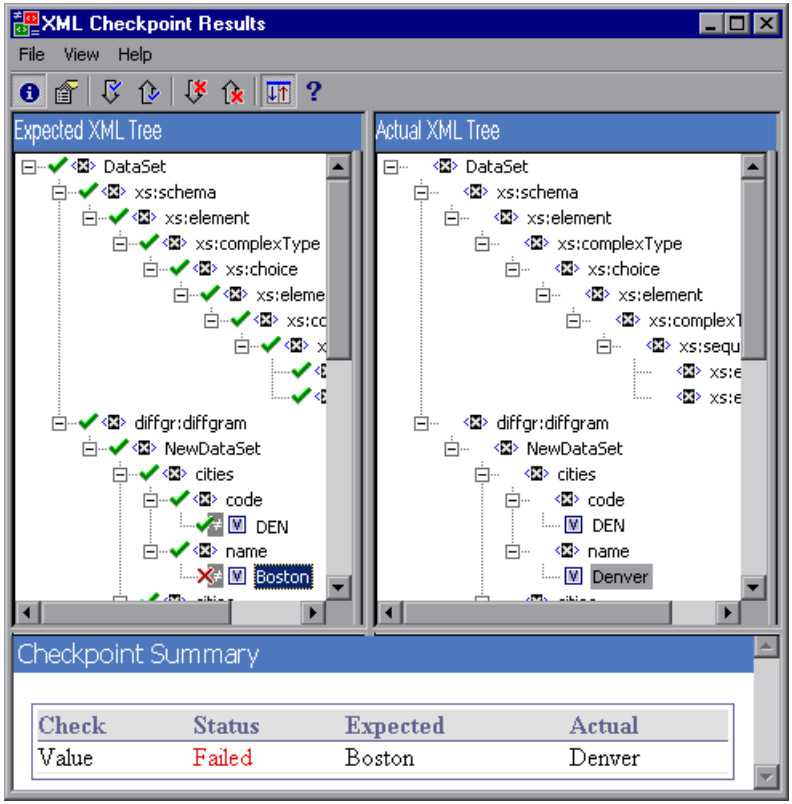
When you click the **View XML Checkpoint Results** button from the Test Results window, the XML Checkpoint Results window displays the XML file hierarchy.

The Expected XML Tree pane displays the expected results—the elements, attributes, and values, as stored in your XML checkpoint.

The Actual XML Tree pane displays the actual results—what the XML document actually looked like during the run session.

The Checkpoint Summary pane displays results information for the check performed on the selected item in the expected results pane.

When you open the XML Checkpoint Results window, the Checkpoint Summary pane displays the summary results for the first checked item in the expected results pane.



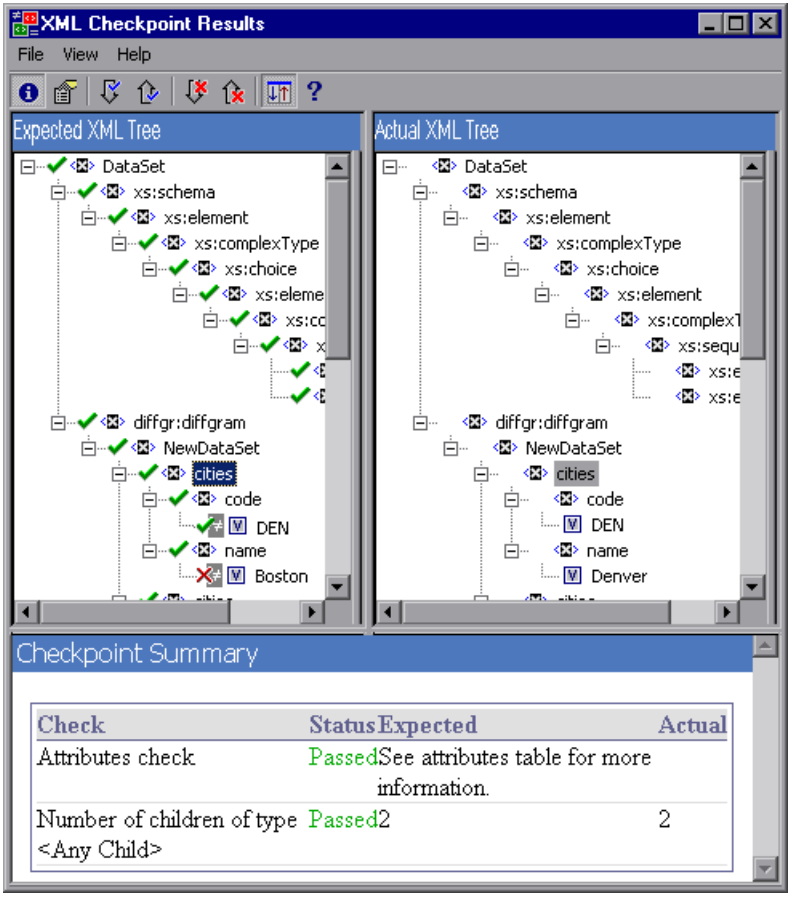
Navigating the XML Checkpoint Results Window

The XML Checkpoint Results window provides a menu and toolbar that enables you to navigate the various parts of your XML checkpoint results.

You can use the commands or toolbar buttons described below to navigate your XML checkpoint results.

➤ **View Checkpoint Summary.** Select an element in the XML Tree and click the **View Checkpoint Summary** button or select **View** > **Checkpoint Summary**. The Checkpoint Summary pane, which provides a detailed description of which parts of an element passed or failed, is displayed at the bottom of the XML Checkpoint Results window.

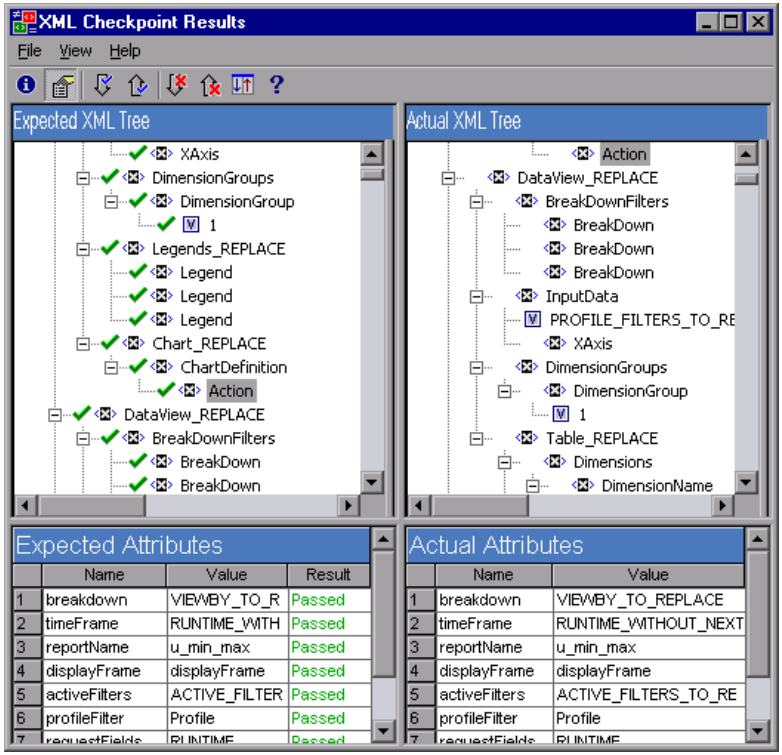
The following example displays the Checkpoint Summary for the cities element in an XML file.



➤ **View Attribute Details.** In the XML Tree, select an element whose attributes were checked. Click the **View Attribute Details** button or select **View > Attribute Details**. Both the Expected Attributes and Actual Attributes panes at the bottom of the XML Checkpoint Results window display the details of the attributes check.

The following example shows the attribute details of the Action element in an XML Web page or frame. The Expected Attributes pane displays each attribute name, its expected value, and the result status of the attribute check.

The Actual Attributes pane displays the attribute name and its actual value during the execution run.



➤ **Find Next Check.** Select **View** > **Find Next Check** or click the **Find Next Check** button to jump directly to the next checked item in the XML Tree.

➤ **Find Previous Check.** Select **View** > **Find Previous Check** or click the **Find Previous Check** button to jump directly to the previous checked item in the XML Tree.

➤ **Find Next Error.** Select **View** > **Find Next Error** or click the **Find Next Error** button to jump directly to the next error in the XML Tree.

➤ **Find Previous Error.** Select **View** > **Find Previous Error** or click the **Find Previous Error** button to jump directly to the previous error in the XML Tree.

➤ **Scroll Trees Simultaneously.** Select **View** > **Scroll Trees Simultaneously**, or click the **Scroll Trees Simultaneously** button to synchronize the scrolling of the Expected and Actual XML Trees. If this option is selected, the Expected and Actual XML Trees scroll simultaneously as you navigate through either of the tree structures. If this option is not selected, you can scroll only one tree at a time.

➤ **View Multi-line Values**. You can double-click any element value in the XML Checkpoint Results window to open the Element Value dialog box, which displays the value in a multi-line edit control. For more information, see “The Element Value Dialog Box” on page 1047.

➤ **Help Topics.** Select **Help** > **Help Topics** or click the **Help Topics** button to view help on the XML Checkpoint Results window.

Examining Sample XML Checkpoint Results

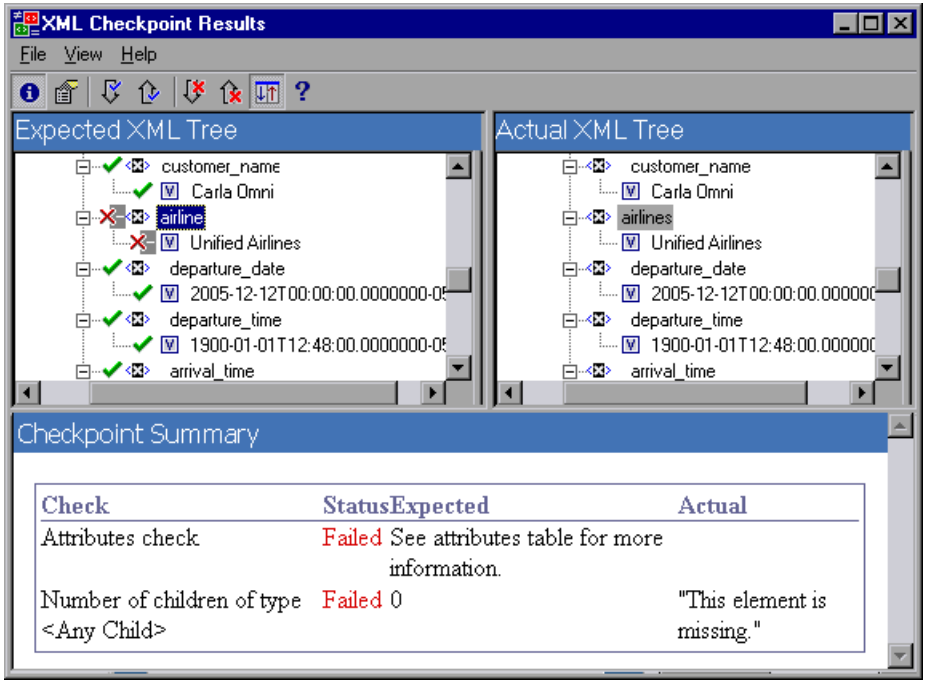
Below are four sample XML checkpoint scenarios. Each example describes the changes that occurred in the actual XML document, explains how you locate the cause of the problem in the XML checkpoint results, and displays the corresponding XML Checkpoint Results window.

Scenario 1

In the following example, the airline element tag was changed to airlines and the XML checkpoint identified the change in the tag structure. The airline element's child element check also failed because of the mismatch at the parent element level.

To view details of the failed element, select the airline tag from the Expected XML Tree and select **View > Checkpoint Summary** to view the Checkpoint Summary in the bottom pane in the XML Checkpoint Results window.

The text "This element is missing" indicates that the airline element tag changed in your XML document.

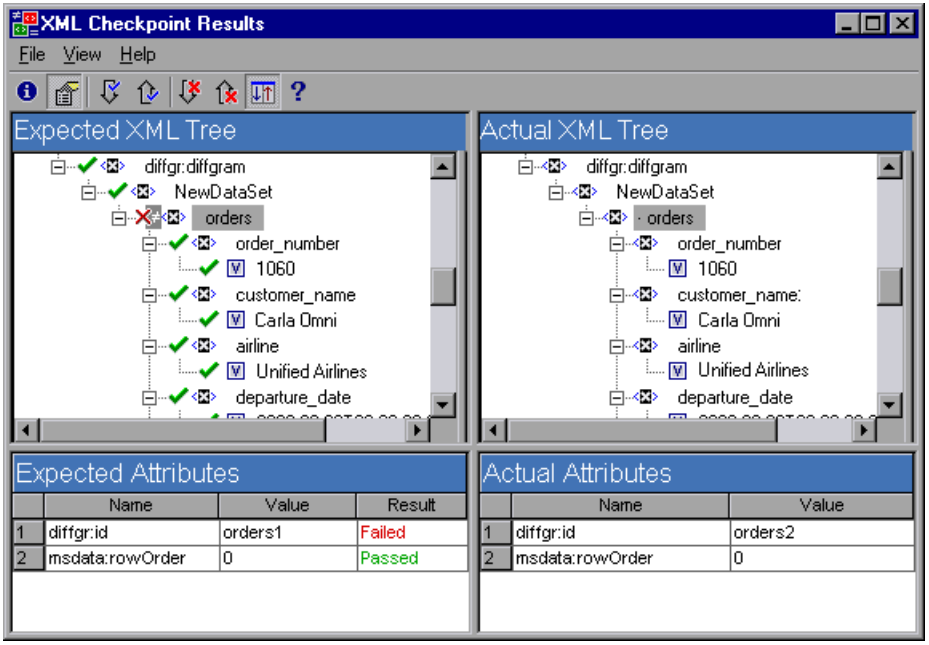


Scenario 2

In the following example, an attribute that is associated with the orders element tag was changed from the original, expected value of orders1, to a new value of orders2.

To view details of the failed attribute, select the failed element from the Expected XML Tree and select **View > Attribute Details**. The Expected Attributes and Actual Attributes panes are displayed at the bottom of the XML Checkpoint Results window.

Using the Expected Attributes and Actual Attributes panes, you can identify which attribute caused the error and which values were mismatched.

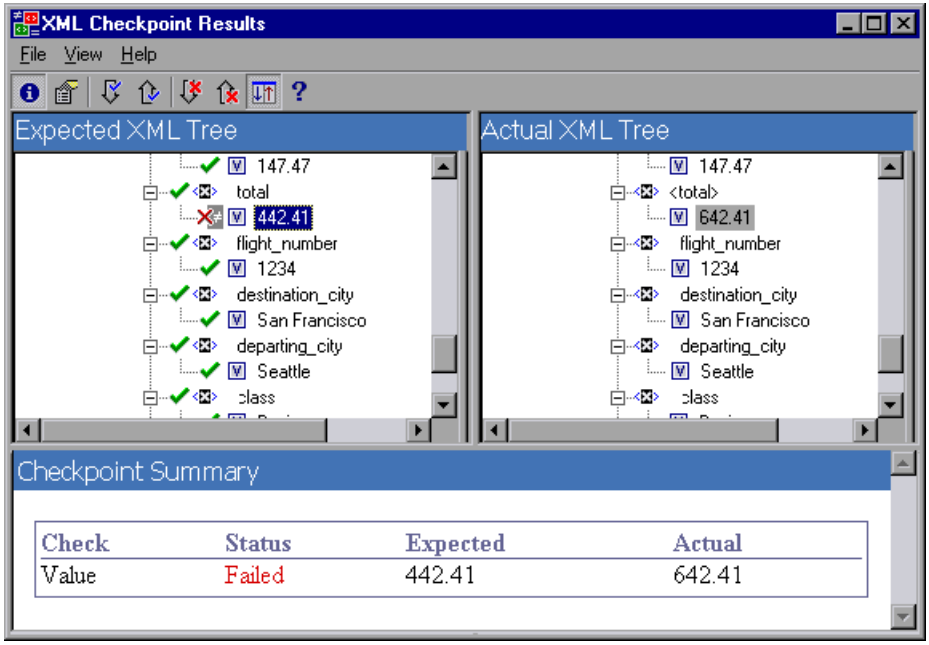


Scenario 3

In the following example, the actual value of the total element was changed between execution runs, causing the checkpoint to fail.

To view details of the failed value, select the failed element from the Expected XML Tree and select **View** > **Checkpoint Summary** to view the Checkpoint Summary in the bottom pane in the XML Checkpoint Results window.

Using the Checkpoint Summary pane, you can compare the expected and actual values of the total element.



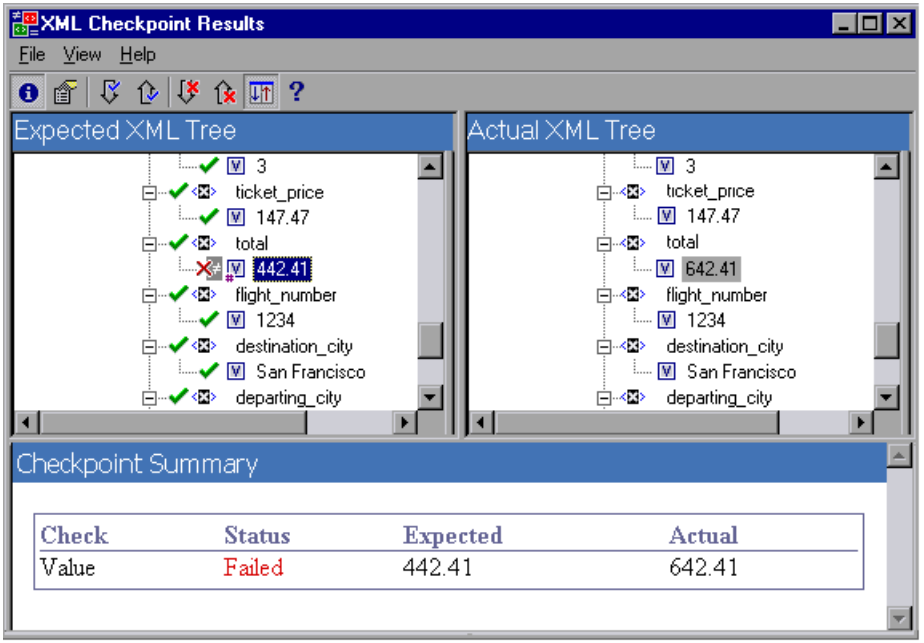
Scenario 4

In the following example, the value of the total element was parameterized and the value's content caused the checkpoint to fail in this iteration.

Note that the value icon is displayed with a pound symbol to indicate that the value was parameterized.

To view details of the failed value, select the failed element from the Expected XML Tree and select **View > Checkpoint Summary** to view the Checkpoint Summary in the bottom pane in the XML Checkpoint Results window. Note that the procedure for analyzing the checkpoint results does not change even though the value was parameterized.

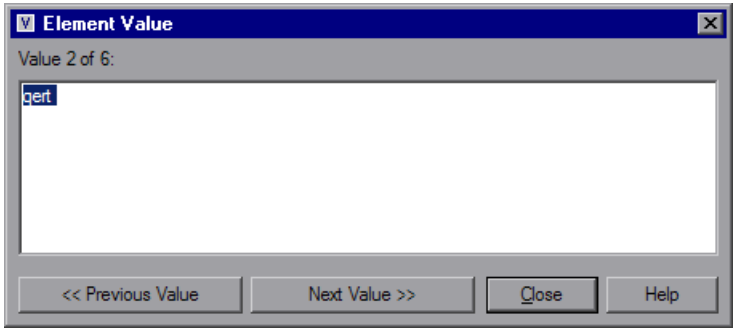
Using the Checkpoint Summary pane, you can compare the expected and actual values of the total element.



The Element Value Dialog Box

Description	Enables you to view element values from the XML Checkpoint Results window in a multi-line edit window. It also enables you to navigate between the values in the **Expected XML Tree** or **Actual XML Tree**.
How to Access	Double-click an element value in the XML Checkpoint Results window.
Learn More	**Conceptual overview:** "Analyzing XML Checkpoint Results" on page 1037.

Below is an image of the Element Value dialog box:



Element Value Dialog Box Options

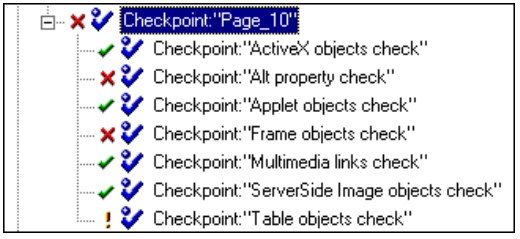
Option	Description
Value x **of** y	Indicates the ordinal position of the selected value within the **Expected XML Tree** or **Actual XML Tree**.
edit window	Displays the full value of the element or attribute in a multi-line window.

Option	Description
Previous Value	Enables you to navigate backward through the element values in the XML Checkpoint Results window. Clicking this button displays the next value in the **Expected XML Tree** or **Actual XML Tree**.
Next Value	Enables you to navigate forward through the element values in the XML Checkpoint Results window. Clicking this button displays the next value in the **Expected XML Tree** or **Actual XML Tree**.

Analyzing Accessibility Checkpoint Results

When you include accessibility checkpoints in your test, the Test Results window displays the results of each accessibility option that you checked.

The run results tree displays a separate step for each accessibility option that was checked in each checkpoint. For example, if you selected all accessibility options, the run results tree for an accessibility checkpoint may look something like this:



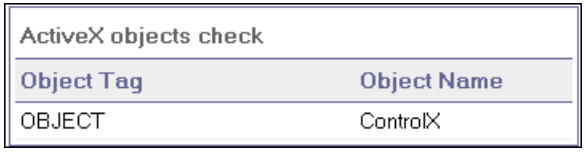
The test result details provide information that can help you pinpoint parts of your Web site that may not conform to the W3C Web Content Accessibility Guidelines. The information provided for each check is based on the W3C requirements.

Note: Some of the W3C Web Content Accessibility Guidelines that are relevant to accessibility checkpoints are cited or summarized in the following sections. This information is not comprehensive. When checking whether your Web site satisfies the W3C Web Content Accessibility Guidelines, you should refer to the complete document at: http://www.w3.org/TR/WAI-WEBCONTENT/.

For more information on accessibility checkpoints, see the section on testing Web objects in the *HP QuickTest Professional Add-ins Guide*.

ActiveX Check

Guideline 6 of the W3C Web Content Accessibility Guidelines requires you to ensure that pages are accessible even when newer technologies are not supported or are turned off. When you select the ActiveX check, QuickTest checks whether the selected page or frame contains any ActiveX objects. If it does not contain any ActiveX objects, the checkpoint passes. If the page or frame does contain ActiveX objects then the results display a warning and a list of the ActiveX objects so that you can check the accessibility of these pages on browsers without ActiveX support. For example:


ActiveX objects check

Object Tag	Object Name
OBJECT	ControlX

Alt Property Check

Guideline 1.1 of the W3C Web Content Accessibility Guidelines requires you to provide a text equivalent for every non-text element. The Alt property check checks whether objects that require the Alt property under this guideline, do in fact have this attribute. If the selected frame or page does not contain any such objects, or if all such objects have the required attribute, the checkpoint passes. If one or more objects that require the property do not have it, the test fails and the test result details display a list that shows which objects are lacking the attribute. For example:

Alt property check

Object Tag	Object Name	Alt Value
IMG	logo	[NONE]
IMG	Dogbert	Dogbert

The bottom pane in the Result Details tab of the Test Results window displays the captured page or frame, so that you can see the objects listed in the Alt property check list.

Applet Check

The Applet Check also helps you ensure that pages are accessible, even when newer technologies are not supported or are turned off (Guideline 6 of the W3C Web Content Accessibility Guidelines), by finding any Java applets or applications in the checked page or frame. The checkpoint passes if the page or frame does not contain any Java applets or applications. Otherwise, the results display a warning and a list of the Java applets and applications. For example:

Applet objects check

Object Tag	Object Name
APPLET	JavaClock.class

Frame Titles Check

Guideline 12.1 of the W3C Web Content Accessibility Guidelines requires you to title each frame to facilitate frame identification and navigation. When you select the Frame Titles check, QuickTest checks whether Frame and Page objects have the TITLE tag. If the selected page or frame and all frames within it have titles, the checkpoint passes. If the page, or one or more frames, do not have the tag, the test fails and the test result details display a list that shows which objects are lacking the tag. For example:

Frame titles check

Object Class	Object Tag	Object Name	Title Value
Frame	IFRAME	takeOver	Takeover Ad
Frame	IFRAME	adSpotFrame5	Click here to find out more!
Frame	IFRAME	theFrame	[NONE]
Page		NBA.com	NBA.com

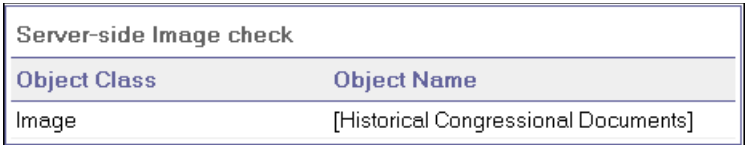
The bottom pane in the Result Details tab of the Test Results window displays the captured page or frame, so that you can see the frames listed in the Frame Titles check list.

Multimedia Links Check

Guidelines 1.3 and 1.4 of the W3C Web Content Accessibility Guidelines require you to provide an auditory, synchronized description of the visual track of a multimedia presentation. Guideline 6 requires you to ensure that pages are accessible, even when newer technologies are not supported or are turned off. The Multimedia Links Check identifies links to multimedia objects so that you can confirm that alternate links are available where necessary. The checkpoint passes if the page or frame does not contain any multimedia links. Otherwise, the results display a warning and a list of the multimedia links.

Server-Side Image Check

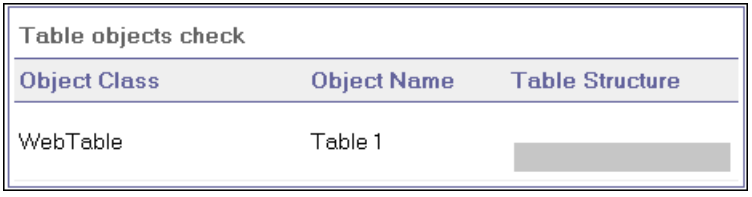
Guideline 1.2 of the W3C Web Content Accessibility Guidelines requires you to provide redundant text links for each active region of a server-side image map. Guideline 9.1 recommends that you provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape. When you select the Server-side Image check, QuickTest checks whether the selected page or frame contains any server-side images. If it does not, the checkpoint passes. If the page or frame does contain server-side images, then the results display a warning and a list of the server-side images so that you can confirm that each one answers the guideline requirements. For example:



Tables Check

Guideline 5 of the W3C Web Content Accessibility Guidelines requires you to ensure that tables have the necessary markup to be transformed by accessible browsers and other user agents. It emphasizes that you should use tables primarily to display truly tabular data and to avoid using tables for layout purposes unless the table still makes sense when linearized. The TH, TD, THEAD, TFOOT, TBODY, COL, and COLGROUP tags are recommended so that user agents can help users to navigate among table cells and access header and other table cell information through auditory means, speech output, or a Braille display.

The Tables Check checks whether the selected page or frame contains any tables. If it does not, the checkpoint passes. If the page or frame does contain tables, the results display a warning and a visual representation of the tag structure of the table. For example:



Viewing Parameterized Values and Output Value Results

You can view information on parameterized values and the results of output value steps in the Test Results window. You can also view the contents of the run-time Data Table.

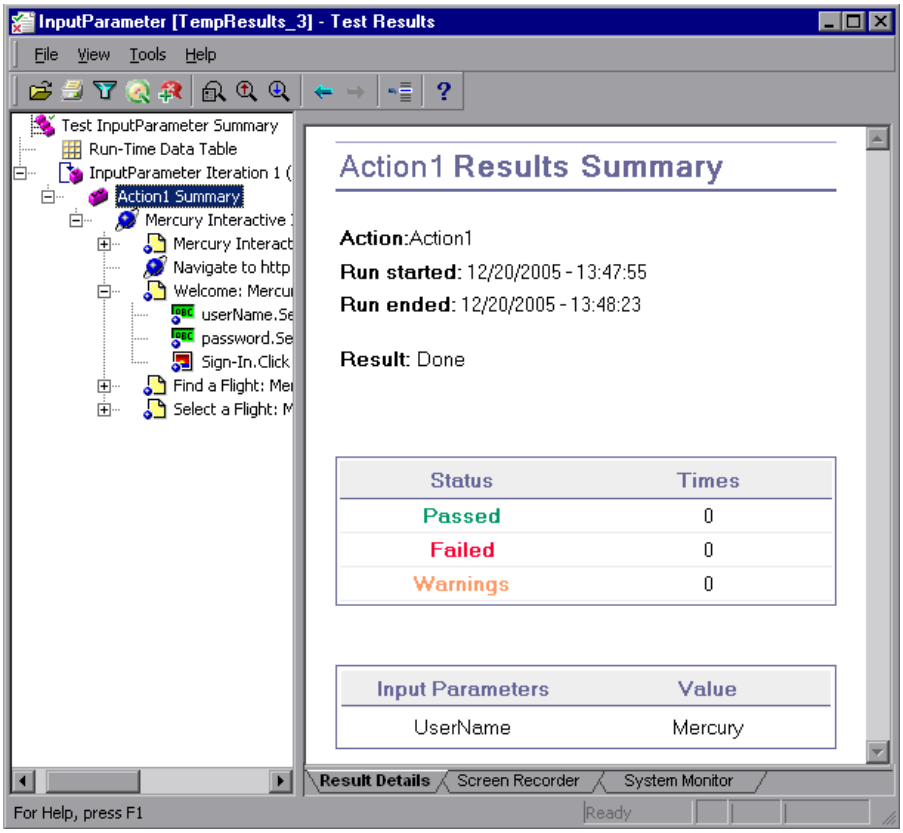
Viewing Parameterized Values in the Test Results Window

A **parameter** is a variable that is assigned a value from an external data source or generator. You can view the values for the parameters defined in your test in the Test Results window.

To view parameterized values:

1. Display the test results for your test in the Test Results window. For more information, see "Viewing the Results of a Run Session" on page 980.
2. In the left pane in the Test Results window, expand the branches of the run results tree and click the branch for the test or action that contains parameterized values.

The name and value of the input parameters are displayed at the bottom of the right pane.



The example above shows the input parameter **UserName** defined for the action with the value **Mercury**.

For more information on defining and using parameters in your tests, see Chapter 24, “Parameterizing Values.”

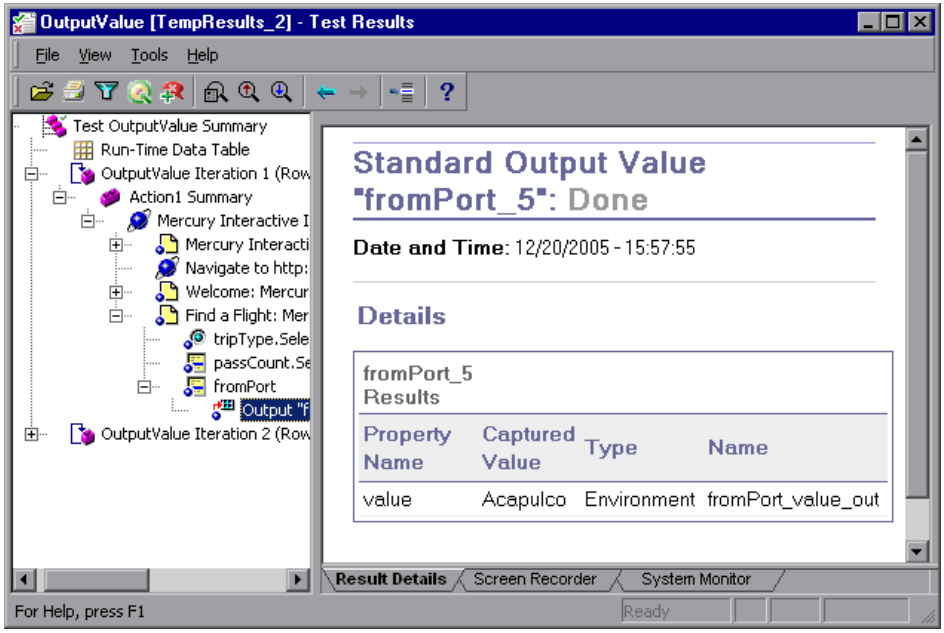
Viewing Output Value Results in the Test Results Window

An **output value** is a step in which one or more values are captured during the run session for use at another point in the run. When one of the values is needed later in the run as input, QuickTest retrieves it from the specified output location.

To view the results of an output value step:

1 Display the test results for your test in the Test Results window. For more information, see "Viewing the Results of a Run Session" on page 980.

2 In the left pane of the Test Results window, expand the branches of the run results tree and click the branch for the output value step whose results you want to view. The output value results are displayed in the Test Results window.



The right pane displays detailed results of the selected output value step, including its status, and the date and time the output value step was run. It also displays the details of the output value, including the value that was captured during the run session, its type, and its name.

For more information on output values, see Chapter 25, “Outputting Values.”

For information on viewing the results of XML output value steps, see “Analyzing XML Output Value Results” on page 1057.

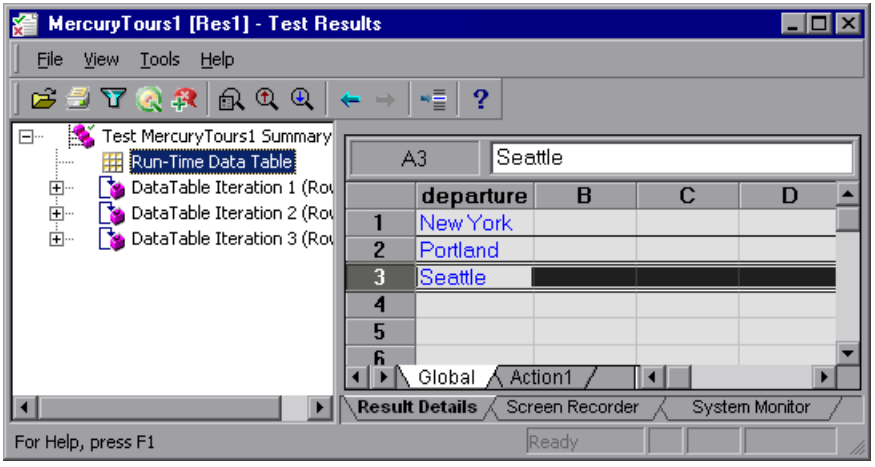
Viewing the Run-Time Data Table

After running a test with Data Table parameters or Data Table output value steps, the Run-Time Data Table displays the parameterized values that were used, as well as any output values stored in the Data Table during the run. You can view the contents of the run-time Data Table in the Test Results window.

To view the run-time Data Table:

1 Display the test results for your test in the Test Results window. For more information, see “Viewing the Results of a Run Session” on page 980.

2 Highlight **Run-Time Data Table** in the left pane in the Test Results window.



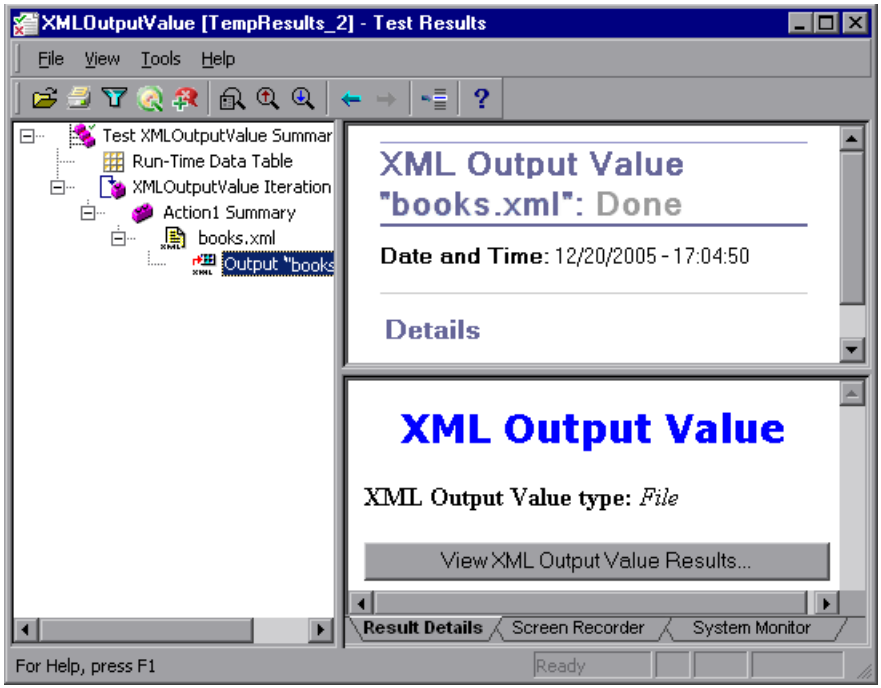
In the above example, the Run-Time Data Table contains the parameterized flight departure values.

For more information on the run-time Data Table, see Chapter 42, “Working with Data Tables.”

Analyzing XML Output Value Results

You can output element or attribute values to your test from XML documents used in your application. For more information on XML output values, see “Outputting XML Values” on page 718.

You can view summary results of the XML output value in the Test Results window. For information on displaying the results for a checkpoint, see “Viewing Checkpoint Results” on page 1028.



The Result Details tab in the right pane displays a summary of the output value results. You can view detailed results by clicking **View XML Output Value Results** to open the XML Output Value Results window.

Note: By default, the **View XML Output Value Results** button is available only when an error occurs. The availability of these detailed results is dependent on the **Save still image captures to results** setting in the Run > Screen Capture pane of the Options dialog box. For more information, see "The Options Dialog Box: Run > Screen Capture Pane" on page 1255.

For more information on XML output value results, see "Understanding the XML Output Value Results Window" on page 1058.

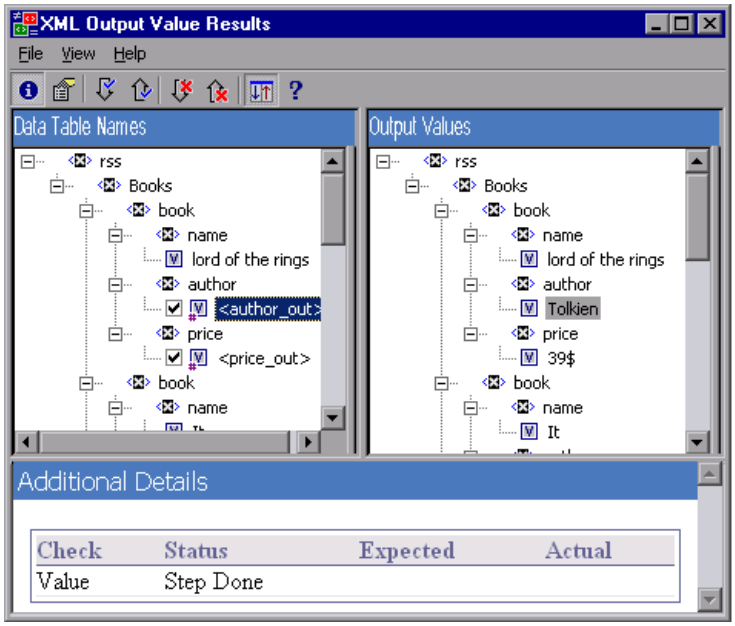
Understanding the XML Output Value Results Window

When you click the View XML Output Value Results button from the Test Results window, the XML Output Value Results window displays the XML file hierarchy.

The Data Table Names pane displays the XML output value settings—the structure of the XML and the Data Table column names you selected to output for Data Table output values.

The Output Values pane displays the actual XML tree—what the XML document or file actually looked like and the actual values that were output during the run.

The Additional Details pane displays results information for the selected item.

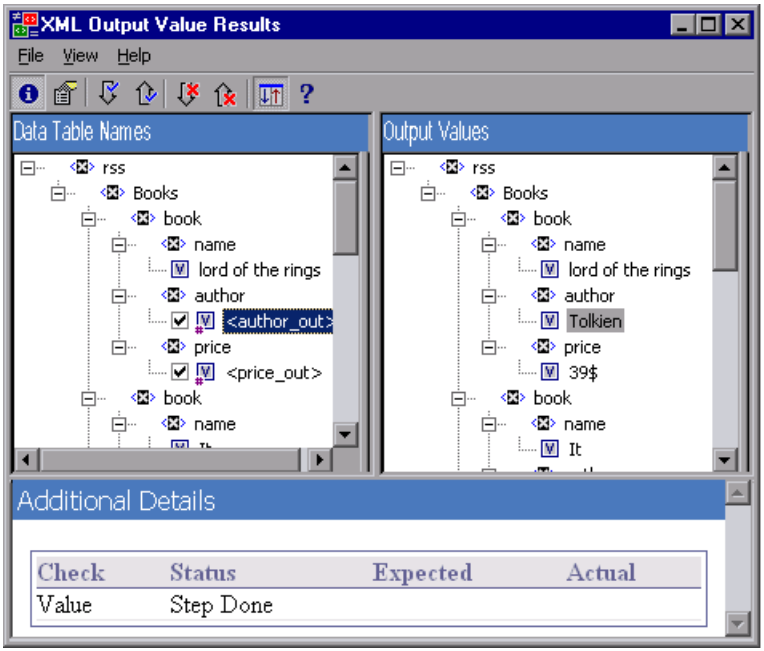


Navigating the XML Output Value Results Window

The XML Output Value Results window provides a menu and toolbar that enables you to navigate the various parts of your XML output value results.

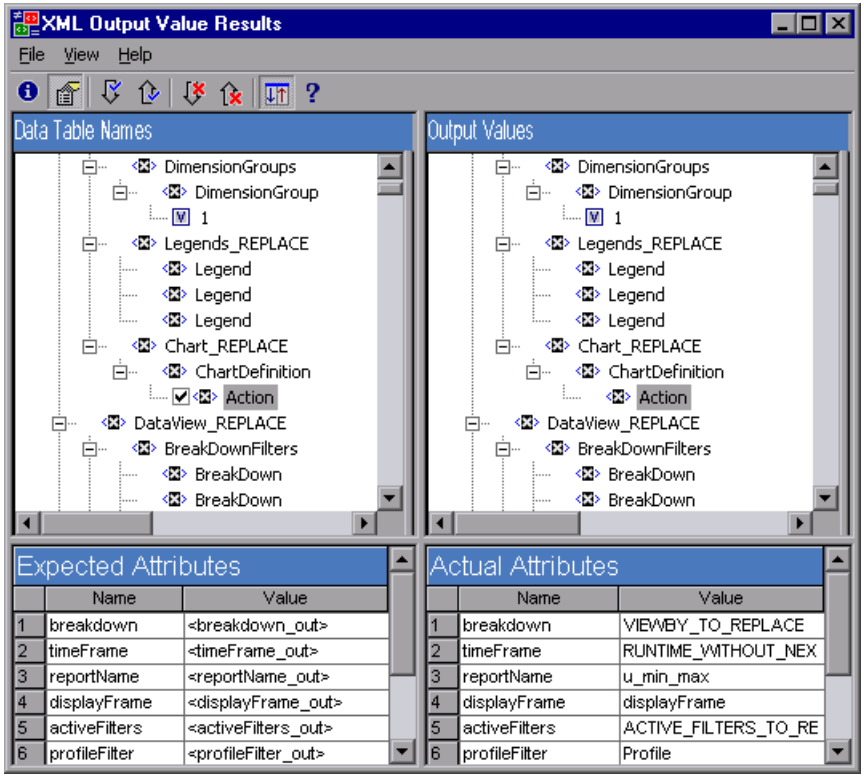
You can use the commands or toolbar buttons described below to navigate your XML output value results:

➤ **View Output Value Summary.** Select an element in the XML Tree and click the **View Output Value Summary** button or select **View** > **Output Value Summary**. The Additional Details pane, which provides information regarding the output value for the selected element, attribute, or value, is displayed at the bottom of the XML Output Value Results window.



➤ **View Attribute Details.** In the XML Tree, select an element whose attributes were output as values. Click the **Attribute Details** button or select **View > Attribute Details**. Both the Expected Attributes and Actual Attributes panes at the bottom of the XML Output Value Results window display the details of the attributes output value.

The Expected Attributes pane displays each attribute name and its expected value or output value name. The Actual Attributes pane displays the attribute name and the actual value of each attribute during the run session.



➤ **Find Next Output Value.** Select **View** > **Find Next Output Value** or click the **Find Next Output Value** button to jump directly to the next output value in the XML Tree.

➤ **Find Previous Output Value.** Select **View** > **Find Previous Output Value** or click the **Find Previous Output Value** button to jump directly to the previous output value in the XML Tree.

➤ **Find Next Error.** Select **View** > **Find Next Error** or click the **Find Next Error** button to jump directly to the next error in the XML Tree.

➤ **Find Previous Error.** Select **View** > **Find Previous Error** or click the **Find Previous Error** button to jump directly to the previous error in the XML Tree.

➤ **Scroll Trees Simultaneously.** Select **View** > **Scroll Trees Simultaneously**, or click the **Scroll Trees Simultaneously** button to synchronize the scrolling of the **Data Table Names** and **Output Values** trees.

If this option is selected, the **Data Table Names** and **Output Values** trees scroll simultaneously as you navigate through either of the tree structures. If this option is not selected, you can scroll only one tree at a time.

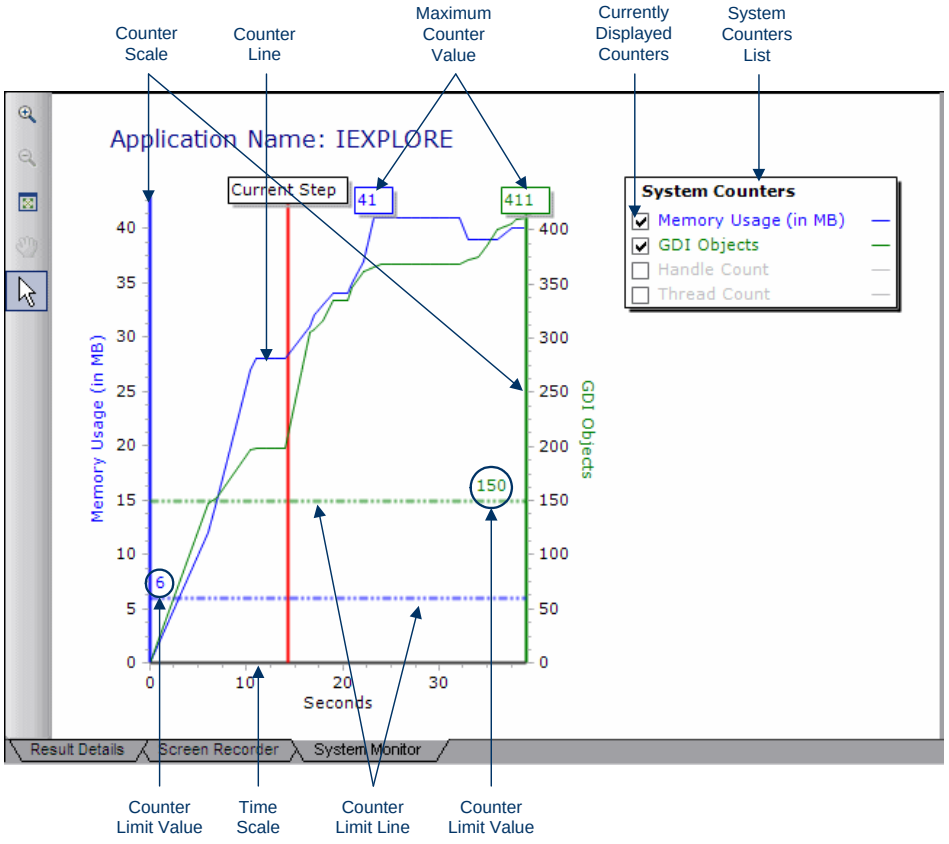
➤ **Help Topics.** Select **Help** > **Help Topics** or click the **Help Topics** button to view help on the XML Output Value Results window.

Viewing System Monitor Results

You view the system counters that you monitored for your test in the System Monitor tab of the Test Results window. Local system monitoring can be enabled for tests only.

For information on enabling local system monitoring for your test, see “Enabling System Monitoring for Your Test” on page 1296.

The System Monitor tab displays the results of the system counters in a line graph.



The System Monitor Tab

The System Monitor tab displays the following information:

- **Application Name**. The name of the application for which system counters were monitored.
- **System Counters List**. The list of system counters monitored for the application.
- **Currently Displayed Counters**. The list of counters currently displayed in the line graph. The System Monitor tab displays a maximum of two counters at one time. To change the counters being displayed, clear the check box for one or both of the currently selected counters, and select the check box for the desired counters.
- **Counter Scale**. The scale of measurement for the performance of that counter.
- **Maximum Counter Value**. The maximum value the counter achieved during the run session.
- **Current Step**. The point in the graph representing the step that is currently highlighted in the Run Results tree.
- **Counter Limit Line**. A visual representation of the limit, if set, for that counter, as defined in the Local System Monitor pane of the Test Settings dialog box. If set, a counter that exceeds this limit causes the step to fail. Only the first step that exceeds the counter limit fails. Subsequent steps that exceed the counter limit are not affected.
- **Counter Limit Value**. The numeric value of the limit, if set, for that counter, as defined in the Local System Monitor pane of the Test Settings dialog box. If set, a counter that exceeds this limit causes the step to fail. Only the first step that exceeds the counter limit fails. Subsequent steps that exceed the counter limit are not affected.
- **Time Scale**. The scale of time in seconds, for the run session.

The System Monitor Tab Colors

Each counter is color coded in the graph. The color of the counter is displayed for:

- the name of the counter in the **System Counters List**
- the **Counter Line**
- the **Counter Scale**
- the **Counter Limit Line**
- the **Counter Limit Value**
- the **Maximum Counter Value**

The System Monitor Tab Toolbar

The System Monitor tab toolbar contains the following buttons:

Button	Usage
	Click the **Zoom In** button and click anywhere on the graph to zoom in. You can also click and drag over an area of the graph to zoom in on that area.
	Click the **Zoom Out** button and click anywhere on the graph to zoom out.
	Click the **View Full Graph** button to zoom out and view the entire graph. This button is disabled when the graph is not zoomed in.
	Click the **Move** button and then click and drag on the graph to scroll right and left. This button is disabled when the graph is not zoomed in.
	Click the **Arrow** button and double-click anywhere on the graph to select that point as the current step. The **Current Step** indicator moves to the new location and the step is highlighted in the Run Results tree. You can also hover over any point on a **Counter Line** in the graph to see the value for the **Counter Line** at that point.

Exporting System Monitor Tab Results

You can export the data from the System Monitor tab to the following file types: **text** (**.csv** or **.txt**), **Excel**, **XML**, or **HTML**.

To export the system monitor data:

Select **File** > **Export System Monitor Data to File** and select a file name and file type for the exported data.

Part VII

Maintaining and Debugging Tests

35

Debugging Tests and Function Libraries

By controlling and debugging your run sessions, you can identify and handle problems in your tests, function libraries, and registered user functions.

This chapter includes:

About Debugging Tests and Function Libraries

After you create a test or function library (including registered user functions), you should check that they run smoothly, without errors in syntax or logic. To debug a function library, you must first associate it with a test and then debug it from that test.

QuickTest provides different options that you can use to detect and isolate defects in a test or function library. For example:

- You can control the run session using the **Pause** command, breakpoints, and various step commands that enable you to step into, over, and out of a specific step.
- If QuickTest displays a run error message during a run session, you can click the **Debug** button on the error message to suspend the run and debug the test or function library.
- When a run session is paused (suspended), you can use the Debug Viewer to check and modify the values of VBScript objects and variables and to manually run VBScript commands.
- You can use the **Debug from Step** command to begin (and pause) your debug session at a specific point in your test. You can also use the **Run to Step** command to pause the run at a specific point in your test. You can set breakpoints, and then enable and disable them as you debug different parts of your test or function library.
- You can also use the **Run from Step** command to run your test from a selected step. This enables you to check a specific section of your application or to confirm that a certain part of your test or function library runs smoothly. For more information, see "Running Part of Your Test" on page 956.

Tip: You can use the Screen Recorder to capture a movie of your application as it is being tested. For more information, see "Viewing Still Images and Movies of Your Application" on page 992.

Considerations for Debugging Tests and Function Libraries

- You must have the Microsoft Script Debugger installed to run tests in debug mode. If it is not installed, you can use the QuickTest Additional Installation Requirements Utility to install it. (Select **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Additional Installation Requirements**.)
- While the test and function libraries are running in debug mode, they are read-only. You can modify the content after you stop the debug session (not when you pause it). If needed, you can enable the function library for editing (**File** > **Enable Editing**) after you stop the session. For more information, see "Editing a Read-Only Function Library" on page 916. After you implement your changes, you can continue debugging your test and function libraries.
- If you perform a file operation (for example, you open a different test or create a new test), the debug session stops.
- If a file is called using an ExecuteFile statement, you cannot debug the file or any of the functions contained in the file. In addition, when debugging a test that contains an ExecuteFile statement, the execution marker may not be displayed correctly.
- In QuickTest, when you open a test, QuickTest creates a local copy of the external resources that are saved to your Quality Center project. Therefore, any changes you apply to any external resource that is saved in your Quality Center project, such as a function library, will not be recognized in the test until the test is closed and reopened. (An external resource is any resource that can be saved separately from the test, such as a function library, a shared object repository, or a recovery scenario.)

  In contrast with this, any changes you apply to external resources saved in the file system, such as function libraries, are implemented immediately, as these files are accessed directly and are not saved as local copies when you open your test.

Slowing a Debug Session

During a run session, QuickTest normally runs steps quickly. While you are debugging a test or function library, you may want QuickTest to run the steps more slowly so you can pause the run when needed or perform another task. You can specify the time (in milliseconds) QuickTest pauses between each step by modifying the **Delay each step execution by** option in the Run pane of the Options dialog box (**Tools** > **Options** > **Run** node). For more information on the Run pane options, see "Setting Run Testing Options" on page 1253.

Using the Single Step Commands

You can run a single step of a test or function library using the **Step Into**, **Step Out**, and **Step Over** commands.

Tip: To display the Debug toolbar, select **View** > **Toolbars** > **Debug**.

Step Into

Step Into runs only the current step in the active test or function library. If the current step calls another action or a function, the called action or function is displayed in the QuickTest window, and the test or function library pauses at the first line of the called action or function.

To use the Step Into command:

Select **Debug** > **Step Into**, click the **Step Into** button, or press F11.

Step Out

After using **Step Into** to enter an action or a function in a function library, you use the **Step Out** command. **Step Out** continues the run to the end of the called action or function, returns to the calling test or function library, and then pauses the run session at the next line (if one exists).

To use the Step Out command:

Select **Debug** > **Step Out**, click the **Step Out** button, or press SHIFT+F11.

Step Over

Step Over runs only the current step in the active test or function library. If the current step calls another action or a user-defined function, the called action or function is executed in its entirety, but the called action or function script is not displayed in the QuickTest window. The run session then returns to the calling test or function library and pauses at the next line (if one exists).

To use the Step Over command:

Select **Debug** > **Step Over**, click the **Step Over** button, or press F10.

Using the Single Step Commands - An Example

Follow the instructions below to create a sample function library and run it (from a test) using the **Step Into**, **Step Out**, and **Step Over** commands.

To create the sample function library and test:	1 Select **File** > **New** > **Function Library** to open a new function library.
	2 In the function library, enter the following lines exactly:
	`public Function myfunc()`
	`msgbox "one"`
	`msgbox "two"`
	`msgbox "three"`
	`End Function`
	3 Save the function library to the file system or your Quality Center project with the name **SampleFL.qfl**. (For more information, see "Saving a Function Library" on page 911.)
	4 Select **File** > **New** > **Test** to open a new test.
	5 Click the tab for the **SampleFL.qfl** function library to bring it into focus.
	6 Select **File** > **Associate Library 'SampleFL.qfl' with 'Test'** to associate the function library with your test.
	7 Click the tab for the test you created to bring it into focus. Click the **Expert View** tab to display the Expert View and enter the following lines exactly:
	`myfunc`
	`myfunc`
	`myfunc`
	`endOfTest="true"`

To run the function library from your test and use the Step Into, Step Out, and Step Over commands:	8 Add a breakpoint on the first step of the test (the first call to the myfunc function) by pressing F9 (**Insert/Remove Breakpoint**). The breakpoint symbol is displayed in the left margin  . For more information, see "Setting Breakpoints" on page 1079. 9 Run the test. The test pauses at the breakpoint. 10 Press F11 (**Step Into**). The execution arrow points to the first line (msgbox "one") of the function in the function library. 11 Press F11 (**Step Into**) again. A message box displays the text one. 12 Click **OK** to close the message box. The execution arrow moves to the next line in the function. 13 Continue pressing F11 (**Step Into**) (and pressing **OK** on the message boxes that open) until the execution arrow leaves the function and is pointing to the second step in the test (the second call to the myfunc function). 14 Press F11 (**Step Into**) to enter the function again. The execution arrow points to the first msgbox line within the function. 15 Press SHIFT+F11 (**Step Out**). Close each of the message boxes that opens. Notice that the execution arrow continues to point to the first line in the function until you close the last of the three message boxes. After you close the third message box, the execution arrow points to the next line in the test (the third call to the myfunc function). 16 Press F10 (**Step Over**). The three message boxes open again—this time, in the Keyword View. The execution arrow remains on the same step in the test until you close the last of the three message boxes. After you close the third message box, the execution arrow points to the next step in the test.

Using the Run to Step and Debug from Step Commands

In addition to stepping into, out of, and over a step while debugging, you can use the **Run to Step** and **Debug from Step** commands to instruct QuickTest to run a test or action (including any associated function library) until it reaches a particular step, or to begin debugging from a specific step.

Run to Step

You can instruct QuickTest to run from the beginning of the test or action (Expert View only)—or from the current location in the test or action—and to stop at a particular step. This is similar to adding a temporary breakpoint to a step. For example, if you want to begin debugging your test or action from a particular step, you may want to run your test or action to that step, as this opens your application to the relevant location.

You can use the **Run to Step** option to start a run session while editing your test or action or to resume a suspended run session.

Do one of the following to instruct QuickTest to run to a particular step:

- In the test, insert your cursor in the step in which you want QuickTest to stop the run and select **Debug** > **Run to Step** or press CTRL+F10.
- In the test, right-click in the step in which you want QuickTest to stop the run and select **Run to Step** from the context menu.
- In the Test Flow pane, right-click the action at which you want QuickTest to stop the run and select **Run to Step** from the context menu. This instructs QuickTest to stop the run at the first step in that action.

Note: If you use the **Run to Step** option to start a new run session, the Run dialog box opens, enabling you to specify the results location and the input parameter values for the debug run session. For more information, see steps 1 and 2 in the "Debug from Step" section, below.

Debug from Step

You can instruct QuickTest to begin your debug session from a particular step instead of beginning the run at the start of the test or action. Before you start debugging from a specific step, make sure that the application is open to the location where you want to start debugging. You can begin debugging from a specific step in your test or action when editing a test or action.

To instruct QuickTest to run from a particular step:

1 Select the step from which you want to begin debugging:

- ➤ Insert your cursor in the step where you want QuickTest to start the run and select **Debug** > **Debug from Step**.
- ➤ Right-click in the step where you want QuickTest to start the run and select **Debug from Step** from the context menu.
- ➤ In the Test Flow pane, right-click the action where you want QuickTest to start the run and select **Debug from Step** from the context menu. This instructs QuickTest to begin the run at the first step in that action.

The Run dialog box opens. For more information on the tabs in the Run dialog box, see "The Run Dialog Box: Results Location Tab" on page 960, and "The Run Dialog Box: Input Parameters Tab" on page 962.

2 If applicable, specify the results location and the input parameter values for the debug run session. By default, the **Temporary run results folder** option is selected.

3 Click **OK**. The Run dialog box closes and the debug run session starts. You can use any of the QuickTest debugging options, such as **Step Into**, **Step Over**, and **Run to Step**.

By default, when the run session ends, the Test Results window opens. For more information on viewing the run results, see Chapter 33, "Viewing Run Session Results." If you cleared the **View results when run session ends** check box in the Run pane of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 44, "Setting Global Testing Options."

Pausing a Run Session

You can temporarily suspend a run session by choosing **Debug** > **Pause** or clicking the **Pause** button. A paused test or function library stops running when all previously interpreted steps have been run.

To resume running a paused run, click the **Run** button, select **Automation** > **Run**, or press **F5**. The run continues from the point it was suspended.

Tip: You can also stop a run session by clicking the **Stop** button, choosing **Automation** > **Stop**, or pressing the Stop command shortcut key (defined in the **Tools** > **Options** > **Run** node). After the run session stops, the Test Results window opens (unless you selected not to view results at the end of a run session (**Tools** > **Options** > **Run** node)).

Using Breakpoints

You can use breakpoints to instruct QuickTest to pause a run session at a predetermined place in a test or function library. QuickTest pauses the run when it reaches the breakpoint, before executing the step. You can then examine the effects of the run up to the breakpoint, make any necessary changes, and continue running the test or function library from the breakpoint. Breakpoints are applicable only to the current QuickTest session and are not saved with your test or function library.

You can use breakpoints to:

- suspend a run session and inspect the state of your application
- mark a point from which to begin stepping through a test or function library using the step commands

You can set breakpoints, and you can temporarily enable and disable them. After you finish using them, you can remove them from your test or function library.

Setting Breakpoints

By setting a breakpoint, you can pause a run session at a predetermined place in a test or function library. A breakpoint is indicated by a filled red circle icon in the left margin adjacent to the selected step.

To set a breakpoint perform one of the following:

- Click in the left margin of a step in the test or function library where you want the run to stop.
- Click a step and then perform one of the following:

  - Click the **Insert/Remove Breakpoint** button.
  - Select **Debug** > **Insert/Remove Breakpoint**.
  - Select **Debug** > **Enable/Disable Breakpoint**.
  - Press F9.

The breakpoint symbol is displayed in the left margin adjacent to the selected step.

Enabling and Disabling Breakpoints

You can instruct QuickTest to ignore an existing breakpoint during a debug session by temporarily disabling the breakpoint. Then, when you run your test or function library, QuickTest runs the step containing the breakpoint, instead of stopping at it. When you enable the breakpoint again, QuickTest pauses there during the next run. This is particularly useful if your test or function library contains many steps, and you want to debug a specific part of it.

You can enable or disable breakpoints individually or all at once. For example, suppose you add breakpoints to various steps throughout your test or function library, but for now, you want to debug only a specific part of your testing document. You could disable all breakpoints in your test or function library, and then enable breakpoints only for specific steps. After you finish debugging that section of your document, you could disable the enabled breakpoints, and then enable the next set of breakpoints (in the section you want to debug). Because the breakpoints are disabled and not removed, you can find and enable any breakpoint, as needed.

Enabled breakpoint. An enabled breakpoint is indicated by a filled red circle icon in the left margin adjacent to the selected step.

Disabled breakpoint. A disabled breakpoint is indicated by an empty circle icon in the left margin adjacent to the selected step.

To enable/disable a specific breakpoint:

1. Click in the step containing the breakpoint you want to disable/enable.
2. Select **Debug** > **Enable/Disable Breakpoint** or press CTRL+F9. The breakpoint is either disabled or enabled (depending on its previous state).

To enable/disable all breakpoints:

Select **Debug** > **Enable/Disable All Breakpoints** or click the **Enable/Disable All Breakpoints** button. If at least one breakpoint is enabled, QuickTest disables all breakpoints in the test or function library. Alternatively, if all breakpoints are disabled, QuickTest enables them.

Removing Breakpoints

You can remove a single breakpoint or all breakpoints defined for the current test or function library.

To remove a single breakpoint perform one of the following:

- Click the breakpoint icon in the left margin of the step.
- Click the step in your test or function library with the breakpoint symbol and:

  - Click the **Insert/Remove Breakpoint** button.
  - Select **Debug** > **Insert/Remove Breakpoint**.
  - Press F9.

The breakpoint symbol is removed from the left margin of the testing document.

To remove all breakpoints:

Click the **Clear All Breakpoints** button, or select **Debug** > **Clear All Breakpoints**. All breakpoint symbols are removed from the left margin of the testing document.

The Debug Viewer Pane

Description	Enables you to perform one of the following activities when a run session is suspended: ➤ View, set, or modify the current value of objects or variables in your test or function library. ➤ Run VBScript commands in your paused run session.
How to Access	Select the **View** > **Debug Viewer** menu command.
Important Information	A run session can be suspended in the following situations: ➤ The run session stops at a breakpoint. ➤ You use **Debug** menu commands or toolbar buttons (such as **Pause** or **Run to Step**) to suspend the run session. ➤ A step fails and you select the **Debug** option.
Learn More	**Conceptual overview**: "About Debugging Tests and Function Libraries" on page 1070 **Primary task**: "Practicing Debugging an Action or a Function" on page 1096

Debug Viewer Pane Tabs

The Debug Viewer pane includes the following tabs:

- **Watch tab.** Displays the current values and types of variables and VBScript expressions that you add to the Watch tab, and enables you to modify the values of displayed variables and properties. For more information, see "The Debug Viewer Pane: Watch Tab" on page 1083.
- **Variables tab.** Displays the current values and types of all variables in the main script of the current action, or in a selected subroutine, and enables you to modify their values. For more information, see "The Debug Viewer Pane: Variables Tab" on page 1089.
- **Command tab.** Enables you to run VBScript commands in your paused run session. For more information, see "The Debug Viewer Pane: Command Tab" on page 1092.

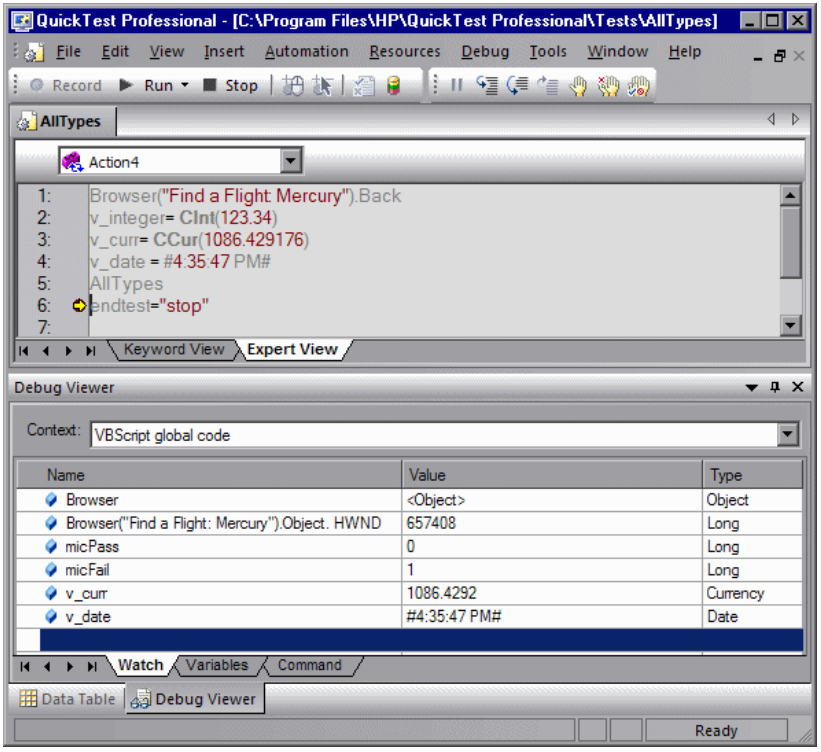
The Debug Viewer Pane: Watch Tab

Description	When a run session is suspended, this tab enables you to view the current values and types of selected variables, properties, and VBScript expressions in your test or function library. You can also use this tab to manually change the value of a variable or property.
How to Access	**View** menu > **Debug Viewer** item > **Watch** tab
Learn More	**Primary task**: "Using the Watch Tab in the Debug Viewer Pane" on page 1087 **Additional related topics**: ➤ "The Debug Viewer Pane" on page 1082 ➤ "Practicing Debugging an Action or a Function" on page 1096

Below is an image of the Watch tab in the Debug Viewer pane:

This image shows a run session that was suspended before running a test step. The **Context** box therefore contains the string VBScript global code and the values displayed in the Watch tab were evaluated within the context of the suspended action.

You can see some of the types of expressions that can be displayed in the Watch tab (for example, the **HWND** native property of the **Find a Flight: Mercury** Browser object). For additional types and contexts, see the image shown in “The Debug Viewer Pane: Variables Tab” on page 1089.



Debug Viewer Watch Tab Details

Item	Description
Context box	Indicates the context in which the expressions displayed in the Watch tab are evaluated. ➤ If the run session was suspended before running a test step, the **Context** box contains the string VBScript global code and the expressions displayed in the Watch tab are evaluated within the context of the suspended action. ➤ If the run session was suspended within a function library, the **Context** box initially displays the name of the function in which the run paused and enables you to switch to the context of other functions and subroutines within the same function library. The expressions displayed in the Watch tab are evaluated within the context of the selected function or subroutine.
Name column	The VBScript expression whose value you want to watch. For information on adding and removing expressions from the Watch tab, see "Using the Watch Tab in the Debug Viewer Pane" on page 1087. **Warning:** QuickTest runs the expressions in the Watch tab to evaluate them. Therefore, do not enter a test object method or any expression whose evaluation could affect the state of the test object, as this can lead to unexpected behavior of your test or function library.

Value column	The current value of the expression. The evaluated value is displayed only when a run session is suspended. In this column, you can also set or modify the value of a variable or property that is being watched. For example, you can edit the value of a run-time object property displayed in the Watch tab, thereby changing the value of the property in the application you are testing before you resume the run session. You cannot modify the run-time value of an object's identification property from the Watch tab.
Type column	The type of the expression's value after it is evaluated (for example, **Integer** or **String**). If an expression cannot be evaluated in the current context, the type displayed is **Error** (indicated also by an icon in the **Name** column).

Using the Watch Tab in the Debug Viewer Pane

You can add VBScript expressions to the Watch tab, to view the current value of different variables and properties of objects in a run session of your test or function library. When the run session is suspended (for example, if you use the **Debug** > **Pause** command, or when the test or function library stops at breakpoint), the Watch tab displays the current values and the types of the expressions that you added to the tab.

As you continue stepping through the subsequent steps in your test or function library, QuickTest automatically updates the Watch tab with the current value for any expression whose value changes.

You can also change the value of a variable or property manually in this tab. For example, you can edit the value of a run-time object property displayed in the Watch tab, thereby changing the value of the property in the application you are testing before you resume the run session. For more information, see "The Debug Viewer Pane: Watch Tab" on page 1083.

Important: QuickTest runs the expressions in the Watch tab to evaluate them. Therefore, do not add a test object method or any expression whose evaluation could affect the state of the test object, as this can lead to unexpected behavior of your test or function library.

You can add any of the following types of expressions to the Watch tab:

- The name of a test object
- The name of a variable
- The name of a property
- Any other type of VBScript expression

Note: To add an **identification property** to the Watch tab, you must use an expression that calls **GetROProperty**. This enables you to watch the run-time value of the object's identification property. For example, to watch the value currently displayed in the Calculator application, you can add the expression: `Window("Calculator").WinEdit("Edit").GetROPRoperty("text")`

You cannot modify the run-time value of an object's identification property from the Watch tab.

To add an expression to the Watch tab:

Perform one of the following:

- Click the expression and select **Debug** > **Add to Watch**.
- Click the expression and press CTRL+T.
- Right-click the expression and select **Add to Watch** from the context menu.
- In the Watch tab, select the empty row in the grid, click in the **Name** column, paste or type the expression, and press ENTER.

Note: You can add an expression to the Watch tab from the Expert View or from a function library.

To remove an expression from the Watch tab:

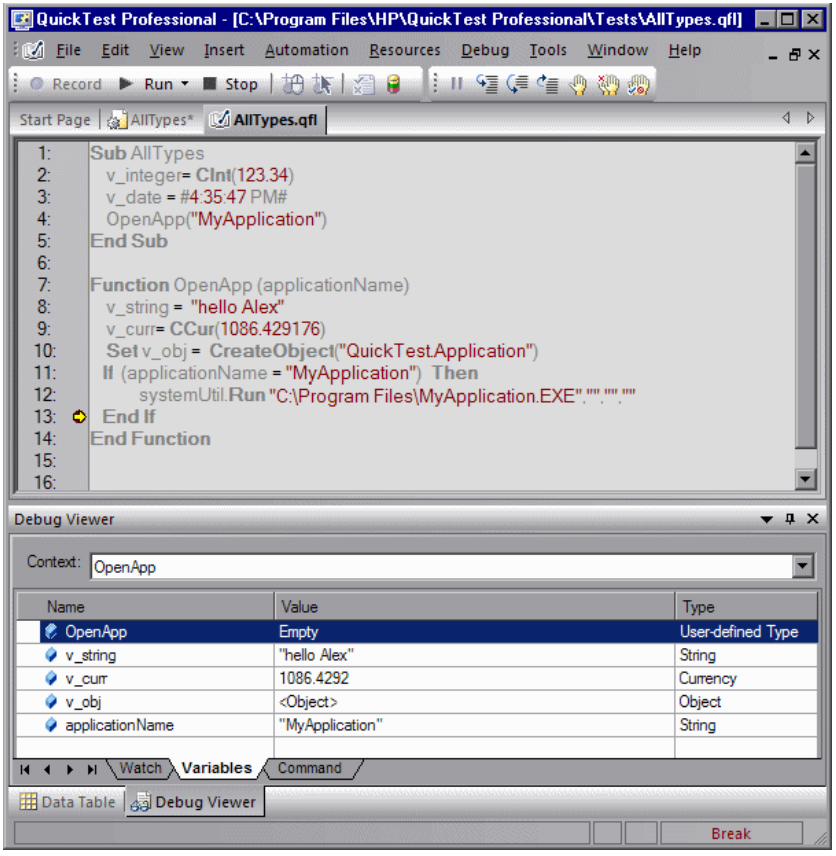
In the Watch tab, select the row that you want to remove and press the Delete key on your keyboard.

The Debug Viewer Pane: Variables Tab

Description	When a run session is suspended, the Variables tab displays the current values and types of all variables in the main script of the current action, or in a selected function in your test or function library, and enables you to modify their values.
How to Access	**View** menu > **Debug Viewer** item > **Variables** tab
Important Information	Only variables that were recognized up to the last step that was performed are displayed in the Variables tab. As you continue stepping through the subsequent steps in your test or function library, QuickTest adds any additional variables that it recognizes and updates the values displayed in the Variables tab.
Learn More	**Additional related topics**: ➤ "The Debug Viewer Pane" on page 1082 ➤ "Practicing Debugging an Action or a Function" on page 1096

Below is an image of the Variables tab in the Debug Viewer pane:

This image shows a run session that was suspended within a function in a function library. The Variables tab therefore displays only the variables that are defined within the context of the suspended function.



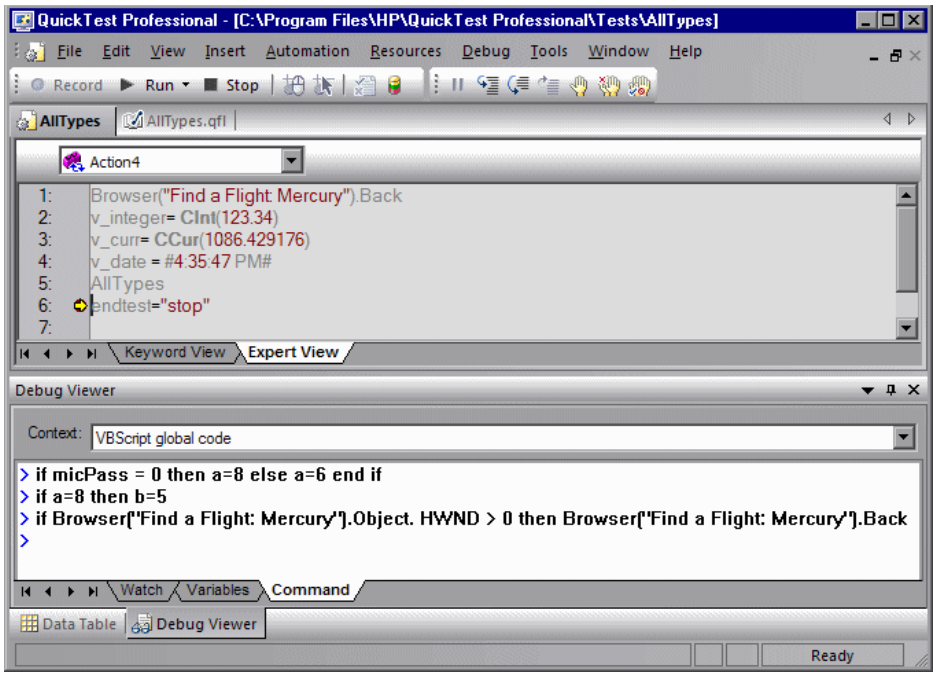
Debug Viewer Variables Tab Details

Item	Description
Context box	Indicates the context of the variables displayed in this tab. ➤ If the run session was suspended before running a test step, the **Context** box contains the string VBScript global code and this tab displays only variables that are defined within the context of the suspended action. ➤ If the run session was suspended within a function library, the **Context** box initially displays the name of the function in which the run paused and enables you to switch to the context of other functions and subroutines within the same function library. Only variables that are defined within the context of the selected function or subroutine are displayed in the Variables tab.
Name column	The name of the variable.
Value column	The current value of the variable. You can edit this value to set or modify the value of the variable before you continue the run session.
Type column	The type of the variable's value (for example, **Integer** or **String**).

The Debug Viewer Pane: Command Tab

Description	When a run session is suspended, this tab enables you to run lines of VBScript code in your test or function library. For example, you can run VBScript code that performs any of the following activities before you resume the run session: ➤ Retrieves information from the application you are testing ➤ Runs a test object method and displays the return value, enabling you to learn more about how the method works ➤ Modifies the value of a native (run-time object) property in the application ➤ Calls a native (run-time object) method in the application
How to Access	**View** menu > **Debug Viewer** item > **Command** tab
Learn More	**Additional related topics**: ➤ "The Debug Viewer Pane" on page 1082 ➤ "Practicing Debugging an Action or a Function" on page 1096

Below is an image of the Command tab in the Debug Viewer pane:



Debug Viewer Command Tab Details

- **Context box.** Indicates the context of the expressions and variables displayed in the Watch and Variable tabs.
- **Command line prompt.** Enables you to run a line of VBScript code in the context of your suspended run session. Type or paste the line of code at the prompt and press ENTER to run the code.
- **Command line history.** Displays the lines of VBScript code that you ran.
  - You cannot make any changes to these lines, but you can select and copy text from them.
  - You can use the UP and DOWN arrow keys to browse through the command history. QuickTest copies the commands to the active command line, enabling you to repeat or reuse commands that you entered earlier.

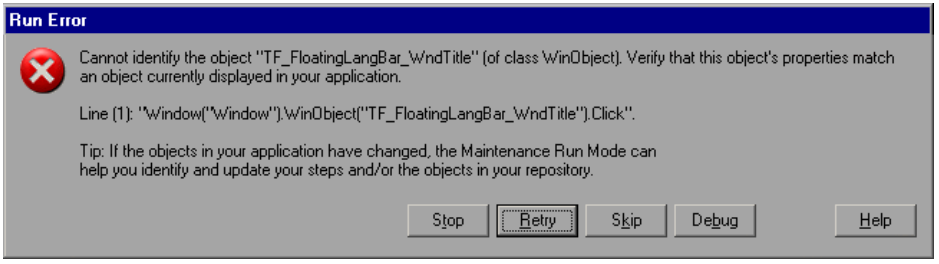
- **Right-click context menu.** Provides commands that you can use to edit the content of the Command tab.
  - The **Cut**, **Copy**, and **Paste** commands enable you to use the clipboard to copy text from the command history and to edit the active command line.
  - The **Clear All** command enables you to erase all of the command history.

Note: You can enter lines of code in the Command tab only when a run session is suspended. When no run session is suspended, you can view the command history, select and copy text from it, or use the **Clear All** context menu command.

Handling Run Errors

There are two types of Run Error message boxes that can be displayed during a run session. One is displayed if the problem is a pure VBScript syntax error. When a syntax run error message box is displayed, click **OK** in the message box and address the error in your step.

The other message box can be displayed in a number of situations. It offers information about the error and a number of buttons for dealing with errors encountered:



- **Stop.** Stops the run session. The run results are displayed if QuickTest is configured to show run results after the run.
- **Retry.** QuickTest attempts to perform the step again. If the step succeeds, the run continues.
- **Skip.** QuickTest skips the step that caused the error, and continues the run from the next step.
- **Debug.** QuickTest suspends the run, enabling you to debug the test and any associated function library that contains a function called by the test.

  You can perform any of the debugging operations described in this chapter. After debugging, you can continue the run session from the step where the test or function library stopped, or you can use the step commands to control the remainder of the run session.
- **Help.** Opens the QuickTest troubleshooting Help for the displayed error message. After you review the Help topic, you can select another button in the error message box.

  The message box also recommends that you consider using Maintenance Mode if you think the error is due to intentional changes in your application and requires that you update multiple steps in your test or objects in your repository. For more information, see "Running Tests with the Maintenance Run Wizard" on page 1104.

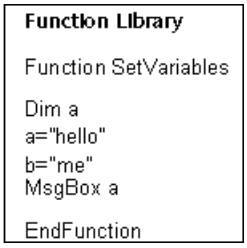
Practicing Debugging an Action or a Function

Suppose you create an action or a function that defines variables that will be used in other parts of your test or function library. You can add breakpoints to the action or function to see how the value of the variables changes as the test or function library runs. To see how the test or function library handles the new value, you can also change the value of one of the variables during a breakpoint.

Step 1: Create a New Action or Function

Open a test and insert a new action, or open a new function library and create a new function called **SetVariables**. For more information on inserting actions, see Chapter 15, "Working with Actions." For more information on working with functions, see Chapter 31, "Working with User-Defined Functions and Function Libraries."

In the Expert View or function library, enter the VBScript code, as follows:



For more information on the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows."

Note: If you are working in the Expert View, skip to Step 4. If you are working in a function library, continue with Step 2 and Step 3.

Step 2: (For Function Libraries Only) Associate the Function Library with a Test

1 Make sure the function library is in focus.

2 Select **File** > **Associate Library** '<**Function Library Name**>' **with** '<**Test Name**>'. QuickTest associates the function library with your test.

Step 3: (For Function Libraries Only) Add a Call to the Function in Your Test

Add a call to the function by inserting a new step and typing the following in the Expert View:

```
SetVariables
```

Step 4: Add Breakpoints

Add breakpoints at the lines containing the text `b="me"` and `MsgBox a`. For more information on adding breakpoints, see "Setting Breakpoints" on page 1079.

Step 5: Begin Running the Test

Run the test. The test or function library stops at the first breakpoint, before executing that step (line of script).

Step 6: Check the Value of the Variables in the Debug Viewer Pane

1 Select **View** > **Debug Viewer** to open the Debug Viewer pane, if it is not already open. Then click the **Watch** tab on the Debug Viewer pane.

2 In the document pane, select the variable **a** and select **Debug** > **Add to Watch**. QuickTest adds the variable a to the Watch tab. The **Value** column indicates that the value of **a** is currently "**hello**", because the breakpoint stopped after the value of variable **a** was initiated. The **Type** column indicates that **a** is a **String** variable.

3 In the document pane, select the variable **b** and select **Debug** > **Add to Watch**. QuickTest adds the variable **b** to the Watch tab. The **Value** column indicates <**Variable is undefined:** '**b**'> (and the **Type** column displays **Error**), because the test stopped before variable **b** was declared.

4 Click the **Variables** tab in the Debug Viewer pane. If you are working with a test, only variable **a** is displayed (with the value "**hello**"), because **a** is the only variable that was initiated up to this point. If you are working with a function library, both **SetVariables** (with the value **Empty**) and variable **a** (with the value "**hello**") are displayed. Variable **b** is not displayed because the test stopped before variable **b** was declared.

Step 7: Check the Value of the Variables at the Next Breakpoint

Click the **Run** button to continue running the test. The test stops at the next breakpoint. Note that the values of variables **a** and **b** were both updated in the Watch and Variables tabs.

Step 8: Modify the Value of a Variable Using the Variables Tab

1 Click the **Variables** tab in the Debug Viewer pane.

2 In the **Value** column, select the string "me", replace it with the string "you", and press ENTER on the keyboard.

3 Click the **Watch** tab. You can see that the value of variable **b** was also updated in the Watch tab.

Step 9: Modify the Value of a Variable Using the Command Tab

1 Click the **Command** tab in the Debug Viewer pane.

2 At the command prompt, type:
`if b="me" then a="b is me" else a="b is you" end if`
Then press ENTER on the keyboard.

3 Click the **Variables** tab to verify that the value of variable **a** was updated according to the command you entered and now displays the value: "b is you"

4 Click the **Run** button to continue running the test. The message box that opens displays "b is you" (which is the modified value of **a**). This indicates that you successfully modified the values of both **a** and **b** using the Debug Viewer pane.

5 Click **OK** to close the message box.

Step 10: Repeat a Command from the Command History

1 Remove the first breakpoint and run the test again. When the test stops at the breakpoint (before displaying the message box), modify the value of variable **b** in the Variables tab to "not me".

2 Select the **Command** tab and press the UP arrow key on your keyboard. QuickTest copies the command that you typed in the previous test run (if b="me" then a="b is me" else a="b is you" end if) to the active command line. Press ENTER to run the command, and then click the **Run** button to complete the test run.

36

Maintaining Tests

QuickTest provides tools that enable you to maintain your tests as the application you are testing changes. For example, your application's objects may change their properties or descriptions, or they may no longer exist. The expected values of your test's checkpoints may also need updating based on changes in your application. This chapter describes how you can use QuickTest's tools to update and maintain your tests.

This chapter includes:

Why Tests Fail

Tests fail when QuickTest encounters a step it cannot perform or the results of a step indicate failure. In many cases this is due to the application being tested not functioning properly. QuickTest then provides you with test results that assist you in understanding how to fix your application.

Sometimes a test fails because the application being tested has changed from when the test was created and the QuickTest test needs to be updated to reflect those changes. Your object repository may also be missing some of the objects it needs to run the test. QuickTest provides tools that help identify and resolve some of these issues.

Object Changes

When QuickTest runs a step in a test, it looks for the object referred to by that step, in the object repositories associated with that test. Using the description of the object in the repository, QuickTest attempts to identify that object in the application.

QuickTest may not be able to identify the object in the application for a number of reasons.

The Object Does Not Exist in the Application

QuickTest cannot find an object in the application that matches the description of the object in the object repository. The Maintenance Run wizard enables you to identify the object that you want your test to use.

The Parent Object Changed

QuickTest cannot find an object in the application that matches and has the same hierarchy as the object in the object repository. The Maintenance Run wizard enables you to identify the object that you want your test to use.

The Object Description Property Values Changed

QuickTest cannot find an object in the application that is similar to, and has the same description property values as the object in the object repository. The Maintenance Run wizard enables you to identify the object that you want your test to use.

The Object Does Not Exist in the Object Repository

QuickTest looks for the object to which the test refers, in the associated object repositories before attempting to identify that object in the application. If the object in your test cannot be found in any associated object repository, The Maintenance Run wizard enables you to identify the object in your application that you want to add to your repository and use in your test.

The Description Set of the Object Needs to Change

QuickTest uses a set of properties to identify objects in the application. If the set of identification properties for the object in the object repository does not provide a unique description matching an object in the application, QuickTest will be unable to find the object. Update Run Mode enables you to update the set of identification properties for the objects in your test to match those defined in the Object Repository dialog box.

Checkpoint Changes

Checkpoints fail when they encounter conditions in the application being tested that are unexpected. In many cases this is due to the application not functioning properly. QuickTest provides you with test results that assist you in understanding how to fix your application.

Sometimes checkpoints fail because the application has changed since the test was created and the QuickTest checkpoints need to be updated to reflect those changes. Update Run Mode enables you to update the checkpoints in your test to reflect changes in the application.

For example, suppose your application has an edit box whose default value used to be <Enter value> and you have checkpoint that checks this value before a new value is entered in the edit box. If the default value in the application changes to be <Enter name> then your checkpoint will fail. Update Run Mode enables you to update the expected values of your checkpoint to reflect the change in the application.

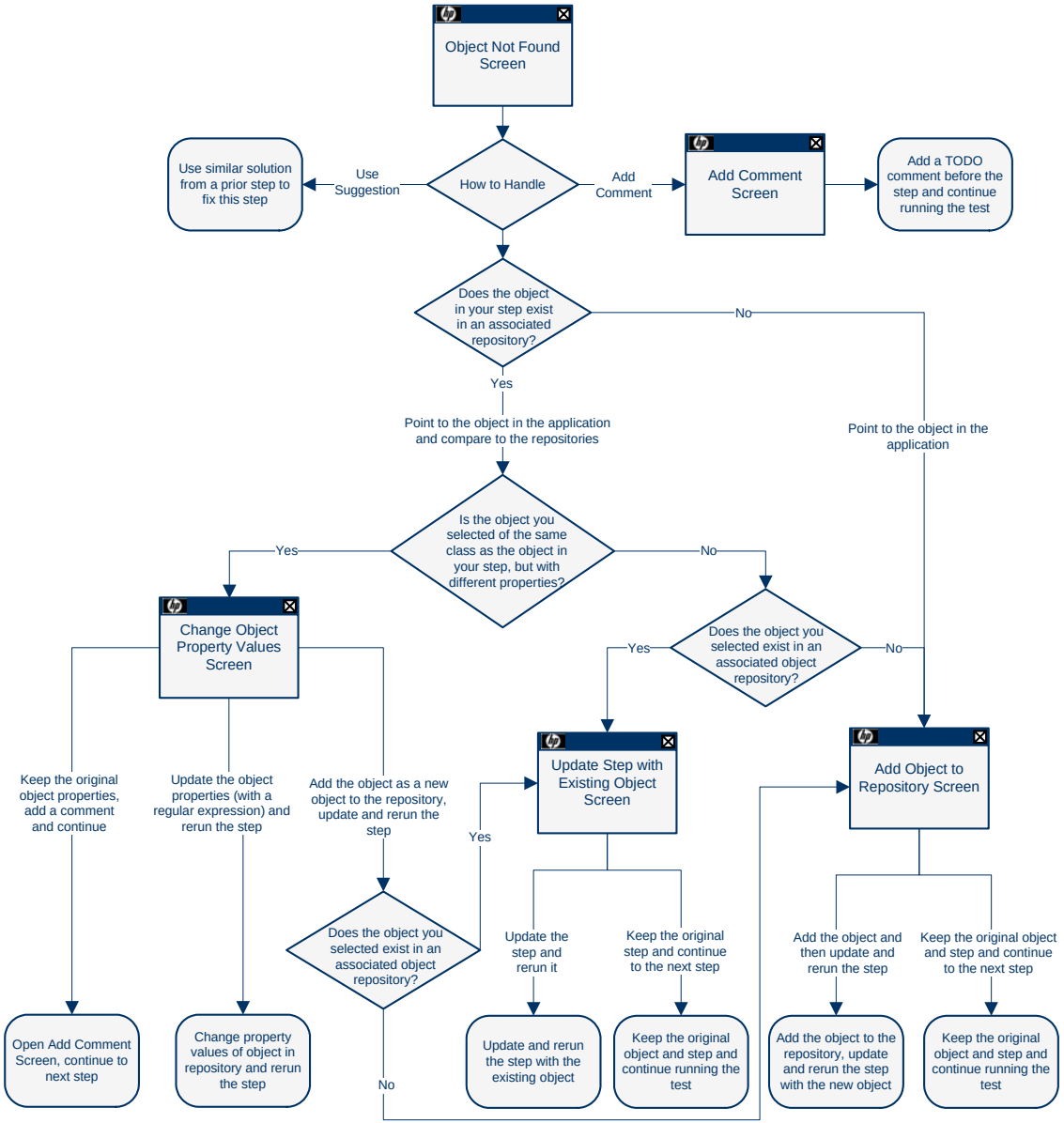
Running Tests with the Maintenance Run Wizard

The Maintenance Run Wizard helps you to maintain your test when it encounters the following problems and provides the following solutions:

Problem	Solution
The step failed because the object in your test cannot be identified in the application.	The Maintenance Run Wizard helps you identify the object in the application that you want your test to use. If you point to an object in the application being tested, the Maintenance Run wizard will compare that object to the objects in the associated object repositories. Depending on how the property values of the object to which you point compare to the property values of the objects in the associated repositories, the Maintenance Run wizard will suggest one of a several options for updating your test to reflect the changes in the application. You can also choose to add a comment to your test before the failed step.
The step failed because the object in your test is missing from your associated object repositories.	The Maintenance Run Wizard helps you add the missing object to the repository. You can also choose to add a comment to your test before the failed step.
The object in your step exists in the application, but can be identified only through Smart Identification.	Identifying objects using Smart Identification may cause tests to run slower. (For more information see, "Configuring Smart Identification" on page 121.) The Maintenance Run Wizard helps you modify the description of the object, so that Smart Identification is not needed.

When you run a test in Maintenance Run Mode, QuickTest runs your test, and then guides you through the process of updating your steps and object repository. The Maintenance Run wizard opens for each of the situations described above. Depending on the problem and user selections, the Maintenance Run wizard will display several screens.

The following flow chart explains the logic of how the wizard and the user determine which screens to display in each situation:



Note: The Object Not Found Screen will not open when QuickTest uses Smart Identification to identify an object in your test. In that case, the Maintenance Run wizard will suggest updating the object properties according to the properties currently defined in the Object Identification dialog box.

When the Maintenance Run Mode ends, the Maintenance Run wizard provides a summary of the changes it made to your test. The main Test Results window also contains a Maintenance Summary which displays details of the changes made to your test, including updated and added objects, updated and commented steps, and a summary of changes to the object repository.

Considerations for Working with the Maintenance Run Wizard

- You must have the Microsoft Script Debugger installed to run the tests in Maintenance Run Mode. If it is not installed, you can use the QuickTest Additional Installation Requirements Utility to install it. (Select **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Additional Installation Requirements**.)
- You can run in Maintenance Run Mode only when QuickTest is set to use the **Normal** (displays execution marker) run mode. It cannot run in **Fast** mode. For more information, see "Setting Run Testing Options" on page 1253.
- You cannot run tests in Maintenance Run Mode on applications that do not have a user interface, such as Web services.
- The Maintenance Run Wizard opens often when your application has changed and QuickTest will be unable to identify objects. You may want to decrease the amount of time QuickTest waits for an object to be displayed before determining that the object cannot be found. You can change the object synchronization timeout in the Run pane of the Test Settings dialog box. Ensure that the timeout specified is sufficient for the objects in your application to load. For more information, see "Defining Run Settings for Your Test" on page 1270.

After Maintenance Run Mode finishes you may want to return this setting to its previous value for regular test runs. (The default QuickTest setting is 20 seconds.)

- As an alternative to using the Maintenance Run wizard, you can update individual test object descriptions from the object in your application using the **Update from Application** option in the Object Repository window or Object Repository Manager. For more information, see "Updating Identification Properties from an Object in Your Application" on page 165.
- After using the Maintenance Run wizard to update the test, you may want to use the **Update from Local Repository** option in the Object Repository Manager to merge the objects from the local repository back to a shared object repository. For more information, see Chapter 7, "Managing Object Repositories."

To run a test in Maintenance Run Mode:

1. Open the test and select **Automation** > **Maintenance Run Mode** or click the down arrow next to the **Run** button in the toolbar and select **Maintenance Run Mode**. The Run dialog box opens.

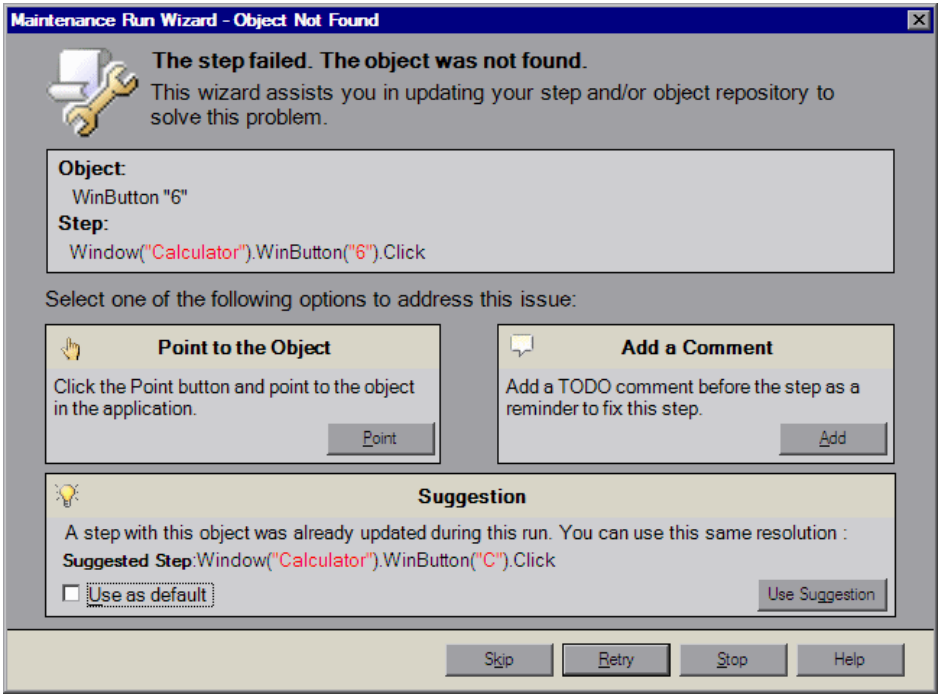
2. Specify the results location and the input parameter values (if applicable) for the Maintenance Run Mode session. For more information, see "The Run Dialog Box: Results Location Tab" on page 960, and "The Run Dialog Box: Input Parameters Tab" on page 962.

3. Click **OK**. The Run dialog box closes and the Maintenance Run Mode session starts.

   By default, when the run session ends, the Test Results window opens. For more information on viewing the run session results, see Chapter 33, "Viewing Run Session Results." If you cleared the **View results when run session ends** check box in the Run pane of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 44, "Setting Global Testing Options."

Maintenance Run Wizard - Object Not Found Screen

If an object in your test cannot be found in the application you are testing or in the associated object repositories, the Object Not Found screen opens. The Object Not Found screen identifies the **Object** that could not be found and the **Step** QuickTest was trying to perform.



Notes:

The **Suggestion** pane is displayed only if the Maintenance Run wizard cannot find an object in the application that was not found earlier in the run session as well.

The **Point to the Object** and **Add a Comment** options are disabled in the Maintenance Run wizard for objects that were not found when:

- The test is open in read-only mode.
- The object is used within a function library function.
- The object's method is defined as a registered user function.

The Object Not Found screen assists you in resolving the problem by providing the following options:

- **Point to the Object.** Click the **Point** button and point to the object in the application that should be used in the step. Use this option if you know the application has changed and identifying a new object for use in the step will resolve the issue, or if the object does not exist in the associated object repositories.

  If the location to which you point is associated with several objects, the Object Selection dialog box opens. Select the correct object from the tree and click **OK**.

  One of the following screens opens depending on the object to which you pointed:

  - "Maintenance Run Wizard - Update Step with Existing Object Screen" on page 1116
  - "Maintenance Run Wizard - Add Object to Repository Screen" on page 1118
  - "Maintenance Run Wizard - Change Object Property Values Screen" on page 1112

- **Add a Comment.** Use this option if you want to add a comment to your test as a reminder to fix the failed step. The Add Comment screen opens.

- **Suggestion.** Displayed only if the Maintenance Run wizard cannot find an object in the application that was not found earlier in the Maintenance Run wizard run as well. If, when the object was first not found, you chose to replace it with a different object, the Maintenance Run wizard will suggest replacing it with the same object now.
  - **Use as default.** If, in subsequent steps the same object cannot be found, the Maintenance Run wizard will automatically replace the object not found with the object you added to the object repository. The Maintenance Run wizard will not open on these subsequent steps.

If you do not use these options, you can use the following buttons to continue:

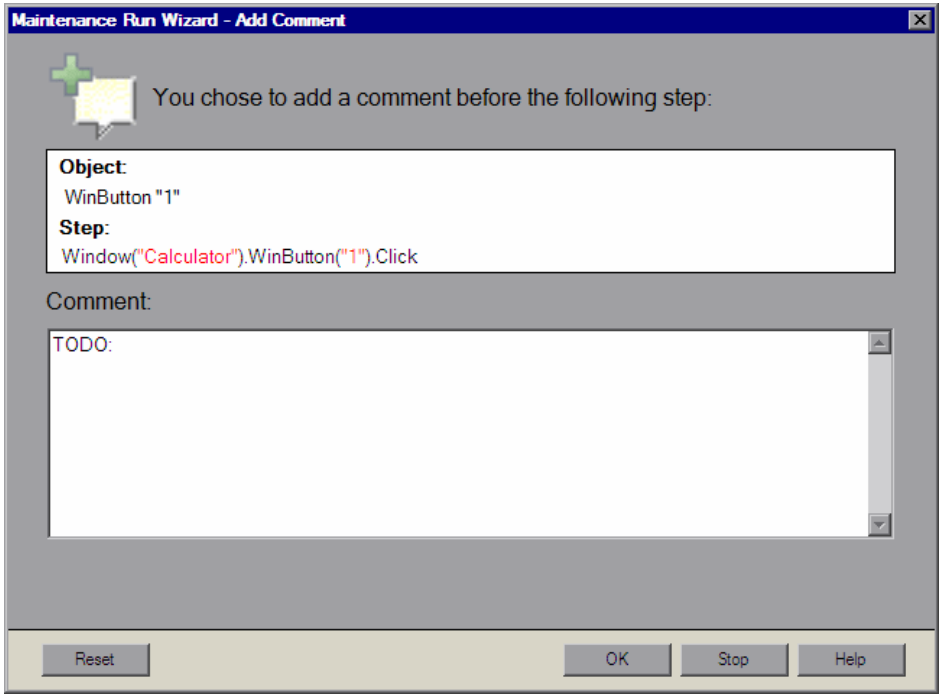
- **Skip.** Skips the current step in the test and continues to run the Maintenance Run wizard on the remainder of the test. This can be used when the problem is in the application being tested and not the QuickTest test.

Note: Before clicking **Skip**, ensure that the application is ready for the next step in the test.

- **Retry.** Retries the current step.
- **Stop.** Stops the Maintenance Run and opens the Maintenance Mode Summary screen.
- **Help.** Opens this Help topic.

Maintenance Run Wizard - Add Comment Screen

The Add Comment screen enables you to add a comment to your test before the current step. This can be used when you believe there is a problem in your test, but identifying the object in the application will not solve the problem, or you want to fix the test manually.

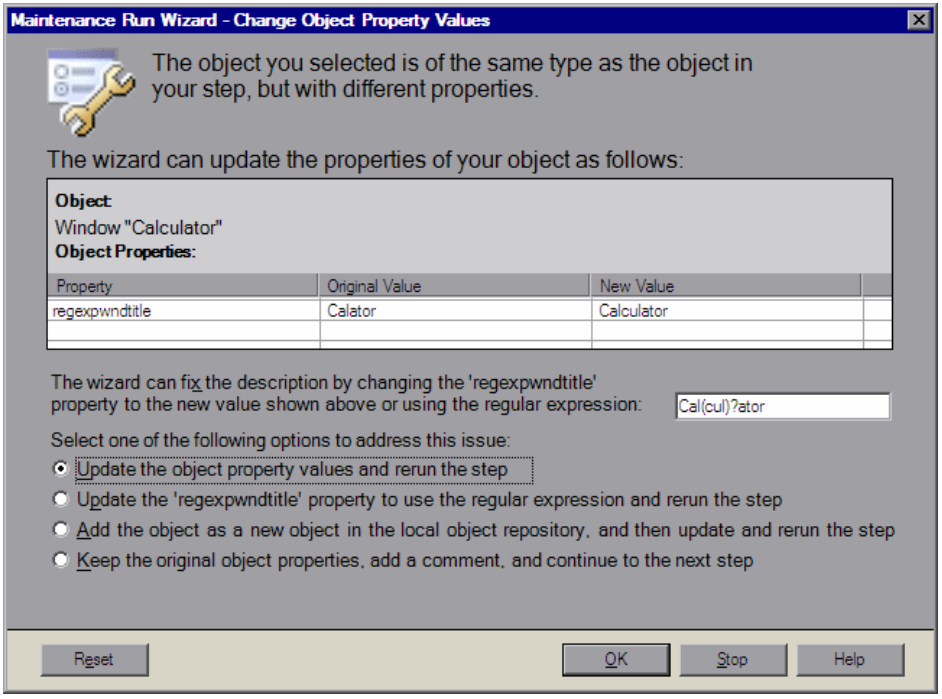


The Add Comment screen creates a comment in your test beginning with the word TODO along with text you add, as a reminder to fix the step at a later time.

Maintenance Run Wizard - Change Object Property Values Screen

The Change Object Property Values screen opens when the object to which you pointed is of the same class as the object in your step, but it has different description property values.

The Change Object Property Values screen suggests updating the property values of the object in the associated object repository to match the property values of the object to which you pointed in the application.



Note: If the Maintenance Run wizard does not determine that a regular expression is relevant for the new property value, the Change Object Property Value screen does not display the suggested regular expression below the properties table. The **Update the <property name> property to use the regular expression and rerun the step** radio button is also not displayed.

The central area of the Change Object Property Values screen contains the following sections:

Section	Description
Object	The object in an associated object repository that is of the same class as the object to which you pointed in the application.
Object Properties	A table displaying the changes that will be made to the property values of the object in the object repository.
Property	The name of the property whose value will be changed.
Original Value	The original property value of the object in the object repository.

Section	Description
New Value	The new property value for the object in the object repository, based on the object to which you pointed in the application.
Recommended regular expression	Depending on the object to which you pointed, the Change Object Property Value screen may include a message that a regular expression can be used to update the property value of the object in the associated object repository. You can modify the suggested regular expression in the edit box. For more information on regular expressions, see "Understanding and Using Regular Expressions" on page 762. **Note:** In a situation where more than one property can use a regular expression, the Maintenance Run wizard will only suggest a regular expression for the first property value.

The Change Object Property Values screen provides the following options:

- **Update the object property and rerun the step.** Updates the property values of the object in the object repository to match those of the object to which you pointed in the application, and reruns the step. The new property values are shown under **New Value**.
- **Update the <property name> property to use the regular expression and rerun the step.** Displayed only if the property value can be updated to use a regular expression. Updates the property value of the object in the object repository with the regular expression as shown in the edit box, and reruns the step.

- **Add the object as a new object in the local object repository, and then update and rerun the step.** This option adds the object to which you pointed, with its current properties, as a new object in the local object repository. This new object may already exist in an associated object repository. One of the following screens opens:
  - The Update Step with Existing Object screen. This screen opens if the object you want to add already exists in an associated object repository.
  - The Add Object to Repository screen. This screen opens if the object you want to add does not already exist in an associated object repository.
- **Keep the original object properties, add a comment, and continue to the next step.** Keeps the original object properties of the object in the object repository. Opens the Add Comment screen, enabling you to add a comment before the step, and then continues to the next step.

The bottom of the screen contains the **Reset** button which enables you to return to the Object Not Found screen, where you can point to a different object in the application or choose a different course of action for this step.

Notes:

- If the object to which you point has a different parent object than the one in the object repository and has different property values, the Change Object Property Values screen opens twice. The first time it enables you to update the parent object of the object in the object repository to match the parent object of the object to which you pointed. The second time it enables you to update the object in the object repository to match the object to which you pointed.
- The Maintenance Run wizard makes changes to the local object repository only. If you want the new object to appear in a shared object repository, use the Object Repository Manager. For more information, see "Performing Merge Operations" on page 240.

Maintenance Run Wizard - Update Step with Existing Object Screen

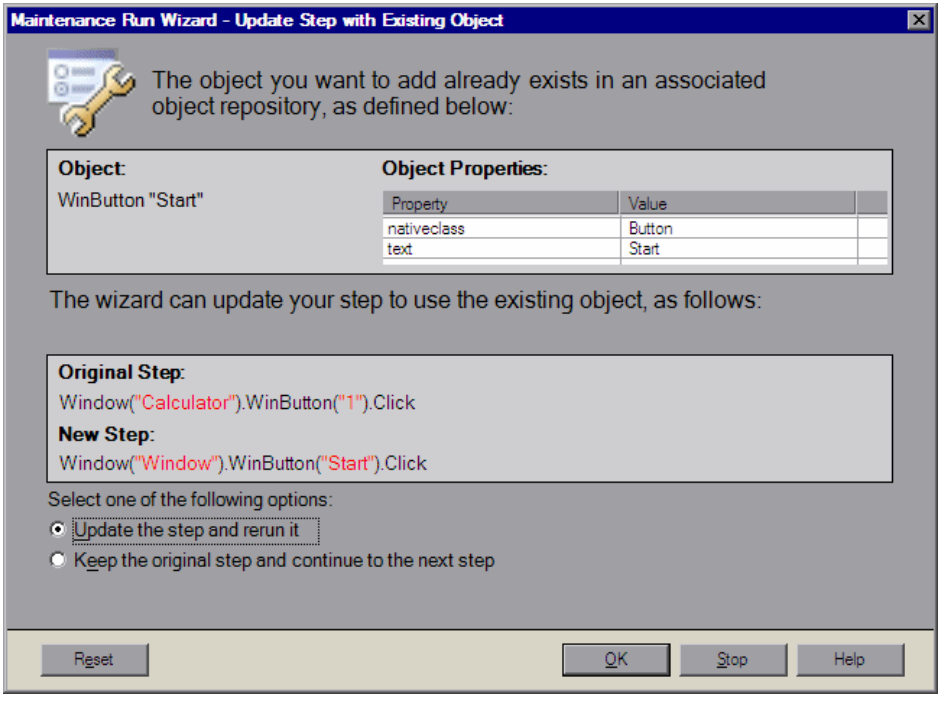
The Update Step with Existing Object screen opens if the object to which you pointed in the Object Not Found screen exists in an associated object repository and:

➤ The object to which you pointed is not of the same class as the object in your step, but with different description property values.

Or

➤ In the Change Object Property Values screen you chose **Add the object as a new object in the local object repository, and then update and rerun the step**.

The Update Step with Existing Object screen suggests updating the step in your test to use an object that already exists in an associated object repository.



The central area of the Update Step with Existing Object screen contains the following sections:

Section	Description
Object	The object in an associated object repository that is the same as the object to which you pointed in the application.
Object Properties	The properties and property values of the object to which you pointed in the test application.
Original Step	The failed original step, with the object that could not be found.
New Step	The new step as it would appear updated to refer to the object which already exists in an associated object repository.

The Update Step with Existing Object screen provides the following options:

➤ **Update the step and rerun it.** Updates the failed step as shown under **New Step** and reruns the step.

Note: The Maintenance Run wizard does not remove the original step from your test. The original step is converted into a comment and the updated step is added below it.

➤ **Keep the original step and continue to the next step.** Keeps the original step and continues to run the Maintenance Run wizard on the remainder of the test.

The bottom of the screen contains the **Reset** button which enables you to return to the Object Not Found screen, where you can point to a different object in the application or choose a different course of action for this step.

Maintenance Run Wizard - Add Object to Repository Screen

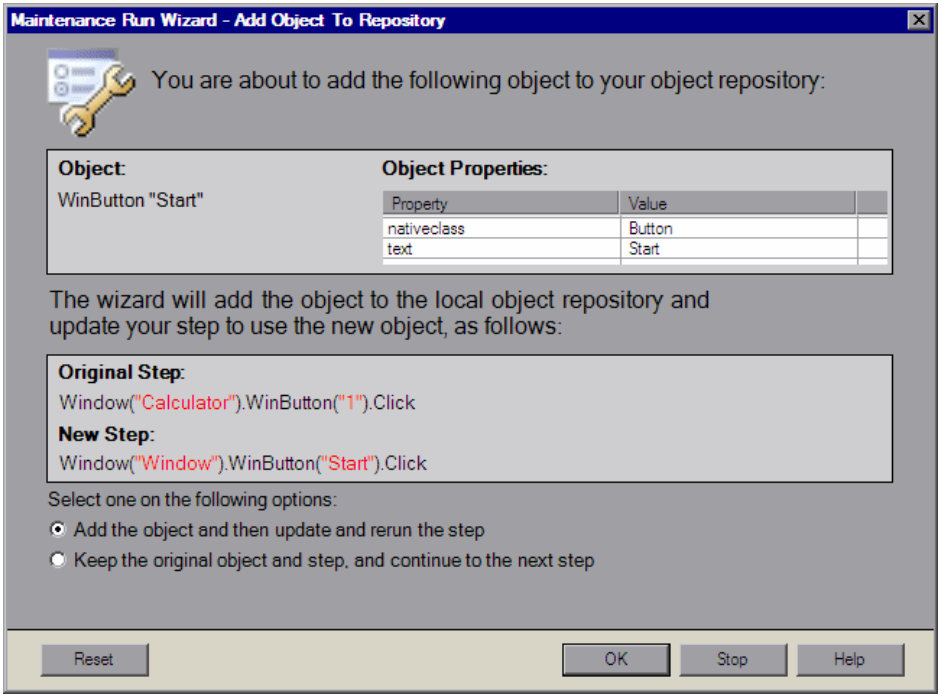
The Add Object to Repository screen opens in the following cases:

- The object **in your step** does not exist in any associated repository.
- The object **to which you pointed** does not exist in any associated object repository and:
  - The object to which you pointed is not of the same class as the object in your step, but with different description property values.

    Or

  - In the Change Object Property Values screen you chose **Add the object as a new object in the local object repository, and then update and rerun the step**.

The Add Object to Repository screen suggests adding the object to which you pointed to the object repository.



The central area of the Add Object to Repository screen contains the following sections:

Section	Description
Object	The object to which you pointed in the test application.
Object Properties	The properties and property values of the object to which you pointed in the test application.
Original Step	The failed original step, with the object that could not be found.
New Step	The new step as it would appear updated to refer to the object being added to the object repository.

The Add Object to Repository screen provides the following options:

- **Add the object and then update and rerun the step.** Adds the new object to the object repository, updates the failed step as shown under **New Step** and reruns the step.
- **Keep the original object and step, and continue to the next step.** Keeps the original step containing the original object and continues to run the Maintenance Run wizard on the remainder of the test.

The bottom of the screen contains the **Reset** button which enables you to return to the Object Not Found screen, where you can point to a different object in the application or choose a different course of action for this step.

Notes:

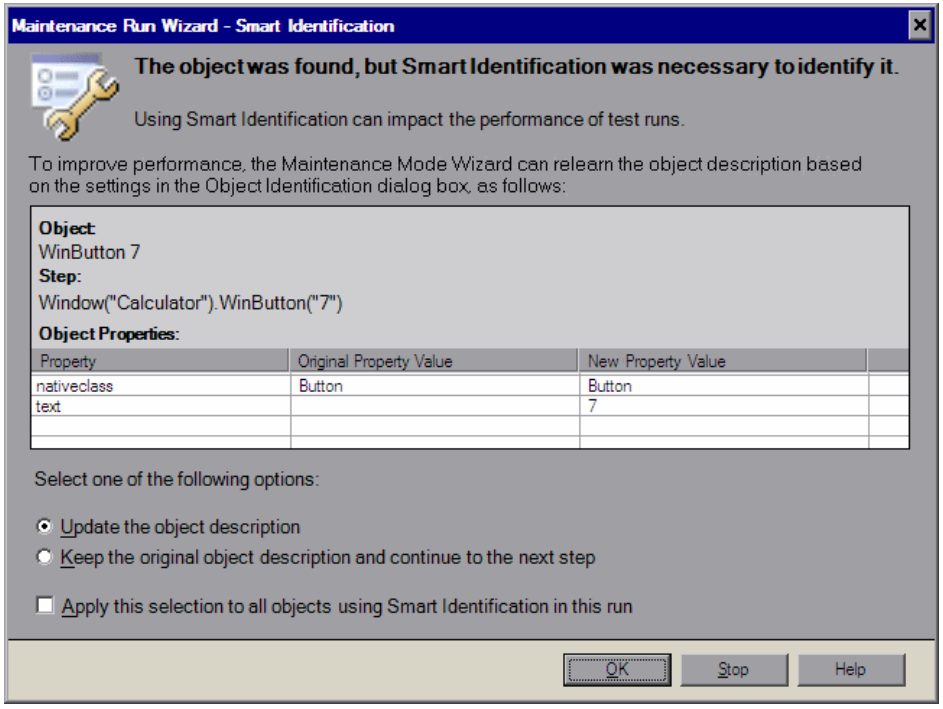
- The Maintenance Run wizard makes changes to the local object repository only. If you want the new object to appear in a shared object repository use the Object Repository Manager. For more information, see "Performing Merge Operations" on page 240.
- The Maintenance Run wizard does not remove the original step from your test. The original step is converted into a comment and the updated step is added below it.

Maintenance Run Wizard - Smart Identification Screen

The Smart Identification screen opens if QuickTest used the Smart Identification mechanism to identify the object in your test. For information on the Smart Identification mechanism, see "Configuring Smart Identification" on page 121.

Smart Identification may slow down test execution, as it is only activated after the object synchronization timeout has been reached.

The Smart Identification screen suggests updating the object description according to the properties currently defined in the Object Identification dialog box.



The central area of the Smart Identification screen contains the following sections:

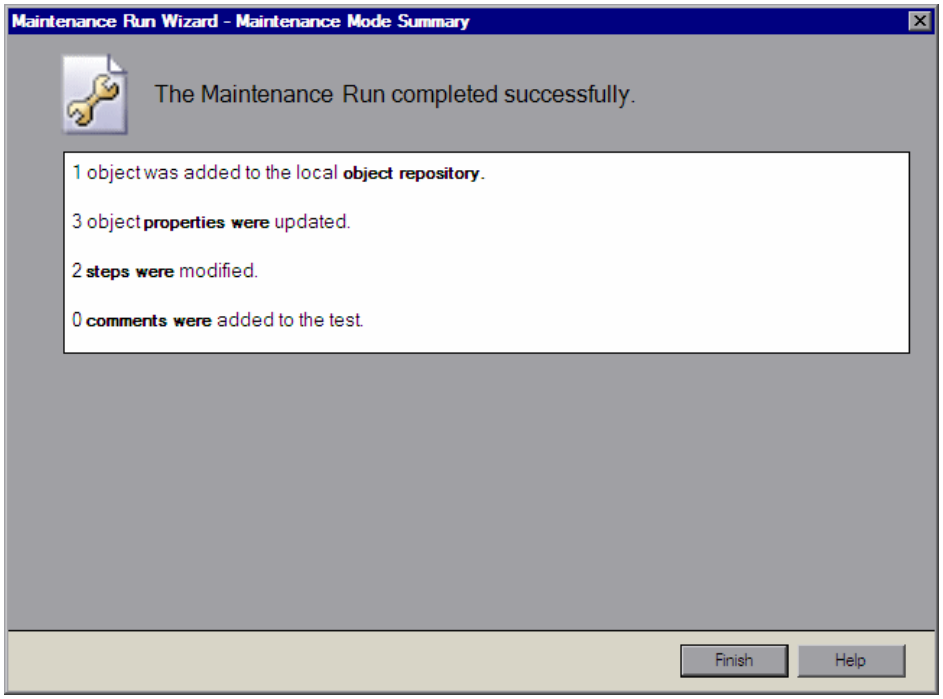
Section	Description
Object	The object in your application that required the Smart Identification mechanism to be identified.
Step	The step in your test in which the object is referenced.
Object Properties	**Property.** The list of properties in the old and new object description. **Original Property Value.** The original value of the property in the **Property** column. Properties that have no value were not part of the original object description. **New Property Value.** The new value of the property in the **Property** column.

The Smart Identification screen provides the following options:

- **Update the object description**. Updates the object description to use the set of properties currently defined in the Object Identification dialog box for the object in your test. Make sure that the set of properties defined in the Object Identification dialog box for the object is sufficient to uniquely identify the object.
- **Keep the original description and continue to the next step**. Keeps the original step containing the original object and continues to run the Maintenance Run wizard on the remainder of the test. The Smart Identification screen will not open for this object again during the run.
- **Apply this selection to all objects using Smart Identification in this run**. Uses your radio button selection above for all objects in the test that need the Smart Identification mechanism to be identified.

Maintenance Run Wizard - Maintenance Mode Summary Screen

When the Maintenance Run wizard is finished, the Maintenance Mode Summary screen opens.



The Maintenance Mode Summary Screen displays the number of objects that were added to the local **object repository**, the number of object **properties** that were updated, the number of **steps** that were modified, and the number of **comments** that were added to the test.

Click **Finish** to end the Maintenance Run wizard. By default, when the run session ends, the Test Results window opens and includes details about the steps and objects that were updated during the run. For more information on viewing the run session results, see "The Test Results Window" on page 971.

Note: If you cleared the **View results when run session ends** check box in the Run pane of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 44, "Setting Global Testing Options."

Updating a Test Using the Update Run Mode Option

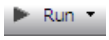
When you run a test in Update Run Mode, QuickTest runs the test to update the test object descriptions, the Active Screen images and values, and/or the expected checkpoint values. After you save the test, the updated data is used for subsequent runs.

When QuickTest updates tests, it runs through only one iteration of the test and one iteration of each action in the test, according to the run option selected. For information on actions, see Chapter 15, "Working with Actions."

Notes:

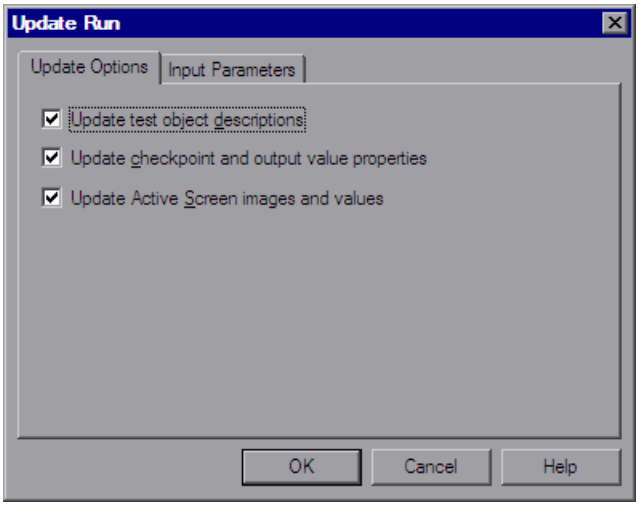
- When a test runs in Update Run Mode, it does not update parameterized values, such as Data Table data and environment variables. For information on parameterized values and environment variables, see Chapter 24, "Parameterizing Values." Update Run Mode does not modify the property values of existing object descriptions in the object repository. To fix the object property values to match your application, use Maintenance Run Mode. For more information, see "Running Tests with the Maintenance Run Wizard" on page 1104.
- When QuickTest updates tests, it always saves the updated objects in the local object repository, even if the objects being updated were originally from a shared object repository. The next time you run the test, QuickTest uses the objects from the local object repository, as the local object repository has a higher priority than any shared object repositories.

Tip: After using **Update Run Mode** to update the test, you may want to use the **Update from Local Repository** option in the Object Repository Manager to merge the objects from the local repository back to a shared object repository. For more information, see Chapter 7, "Managing Object Repositories."



1 Open the test, and select Select **Automation** > **Update Run Mode**, or click the down arrow next to the **Run** button in the toolbar and select **Update Run Mode**.

The Update Run dialog box opens.



2 Specify the settings for the update run process. For more information, see "Understanding the Update Options Tab" on page 1128, and "The Run Dialog Box: Input Parameters Tab" on page 962.

Note: The run results for an update run session are always saved in a temporary location.

---

3 Click **OK**. The Update Run dialog box closes and QuickTest begins running in Update Run Mode. The text **Update Run** flashes in the status bar while the test is being updated.

QuickTest runs the test and updates the test object descriptions, the Active Screen information, and/or the expected checkpoint values, depending on your selections. When the run session ends, the Test Results window opens.

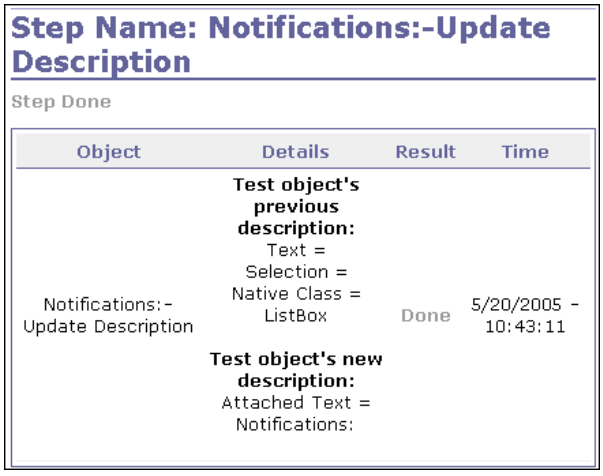
For information on viewing the results, see Chapter 33, "Viewing Run Session Results."

Note: If you cleared the **View results when run session ends** check box in the Run pane of the Options dialog box, the Test Results window does not open at the end of the update run session. For more information on the Options dialog box, see Chapter 44, "Setting Global Testing Options."

When the update run ends, the Test Results window can show:

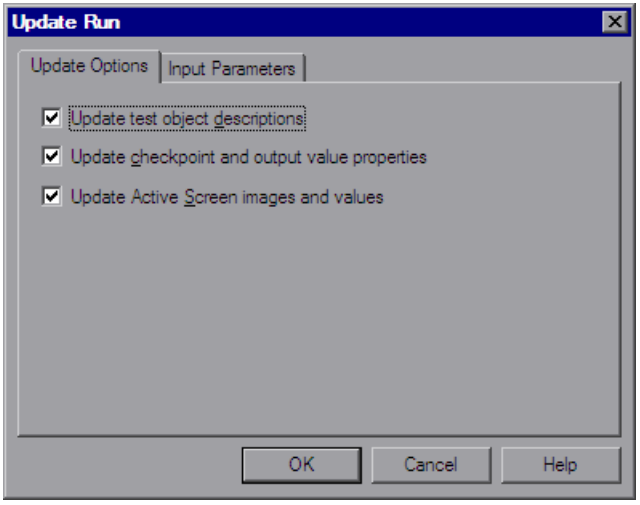
- Updated values for checkpoints.
- Updated test object descriptions.

For example:



Understanding the Update Options Tab

The Update Options tab enables you to specify which aspects of your test you want to update, such as test object descriptions, expected checkpoint values, and/or Active Screen images and values. After you save the test, the results of the updated test are used for subsequent runs.



You can specify one or more of the following information types to update:

➤ **Update test object descriptions.** QuickTest updates the set of properties for each object class in your associated object repositories according to the properties currently defined in the Object Identification dialog box. You can use this option to modify the set of properties used to identify an object of a certain type.

Note: If the property set you select in the Object Identification dialog box for an object class is not ideal for a particular object, the new object description may cause future runs to fail. Therefore, it is recommended that you save a copy of your object repositories (or check them into a Quality Center project with version control support, if applicable) before updating them, so that you can return to the previously saved version, if necessary.

This option can be especially useful when you want to create or debug your test steps using object property values that are easy to recognize in your application (such as object labels), but may be language- or operating system-dependent. After you debug your test, you can use the **Update Run Mode** option to change the object descriptions to use more universal property values.

For example, suppose you design a test for the English version of your application. The test objects are recognized according to the identification property values in the English version, some of which may be language-dependent. You now want to use the same test for the French version of your application.

To do this, you define properties that are non-language-dependent, so that QuickTest can use these properties for object identification. For example, you can identify a link object by its **target** property value instead of its **text** property value. You can then perform an update run on the English version of your application using these new properties. This modifies the test object descriptions so that you can later run the test successfully on the French version of your application.

Tip: If you have a test that runs successfully, but in which certain objects are identified using Smart Identification, you can change the set of properties used for object identification and then use the **Update test object descriptions** option to update the test object description to use the set of properties that Smart Identification used to identify the object.

When you run the test with **Update test object descriptions** selected, QuickTest finds the test object specified in each step based on its current test object description. If QuickTest cannot find the test object based on its description, it uses the Smart Identification properties to identify the test object (if Smart Identification is enabled). After QuickTest finds the test object, it then updates its description based on the mandatory and assistive properties that you define in the Object Identification dialog box.

Note: Test objects that cannot be identified during the update process are not updated. As in any run session, if an object cannot be found during the update run, the run session fails, and information on the failure is included in the Test Results. In these situations, you may want to use Maintenance Run Mode to resolve these problems.

Any properties that were used in the previous test object description and are no longer part of the description for that test object class, as defined in the Object Identification dialog box, are removed from the new description, even if the values were parameterized or defined as regular expressions.

If the same property appears both in the test object's new and previous descriptions, one of the following occurs:

- If the property value in the previous description is parameterized or specified as a regular expression and matches the new property value, QuickTest keeps the property's previous parameterized or regular expression value. For example, if the previous property value was defined as the regular expression button.*, and the new value is button1, the property value remains button.*.
- If the property value in the previous description does not match the new property value, but the object is found using Smart Identification, QuickTest updates the property value to the new, constant property value. For example, if the previous property value was button.*, and the new value is My button, if a Smart Identification definition enabled QuickTest to find the object, My button becomes the new property value. In this case, any parameterization or use of regular expressions is removed from the test object description.

➤ **Update checkpoint properties and output property values.** QuickTest updates the expected values of your checkpoints to reflect any changes that may have occurred in your application since you created the test and updates the list of items that can be retrieved in your output value steps.

For example, suppose you defined a text checkpoint as part of your test, and the text in your application has changed since you created your test. You can update the test to update the checkpoint properties to reflect the new text.

The output value option is mainly relevant for XML output value steps used with Web services test. For more information, see the section describing Web services in the *HP QuickTest Professional Add-ins Guide*.

Notes:

➤ If checkpoint property values are parameterized or include regular expressions, they are not updated when using this option.

➤ If your test includes calls to a WinRunner test and you have write permissions for both the test and the expected results folder, then selecting **Update checkpoint properties** also updates the expected values of the checkpoints in your WinRunner test. If you do not want to update the WinRunner test, you may want to comment out the line that calls the WinRunner test. For more information on calling WinRunner tests, see "Calling WinRunner Tests" on page 1518. For more information on comment lines, see "Adding Comments" on page 815.

➤ If you selected the **Save only selected area** check box when creating a bitmap checkpoint, the **Update Run Mode** option updates only the saved area of the bitmap; it does not update the original, full size object. To include more of the object in the checkpoint, create a new checkpoint. For more information, see "Checking Bitmaps" on page 515.

➤ **Update Active Screen images and values.** QuickTest updates images and property values in the Active Screen to reflect any changes that may have occurred in your application since you recorded the test or if the Active Screen does not appear as it should. For example, suppose a dialog box in your application has changed since you recorded your test. You can update the test to update the dialog box appearance and its properties in the Active Screen.

You can also use this option to increase or decrease the amount of information saved and displayed in your Active Screen. Change the Capture Level slider (**Tools** > **Options** > **Active Screen** node), and run the test in Update Run Mode with the **Update Active Screen images and values** check box selected. QuickTest updates the amount of information it saves and displays in the Active Screen, based on the new setting. For more information, see "Setting Active Screen Options" on page 1240.

Part VIII

Working with the QuickTest IDE

37

QuickTest Window Layout

This chapter describes how to customize the QuickTest window and work with QuickTest documents.

This chapter includes:

Modifying the QuickTest Window Layout

You can modify the layout of the QuickTest window. For example, you can move and resize panes, select to show or auto-hide panes, create tabbed panes, and select which toolbars to display. If needed, you can also restore the default layout.

You can also resize the QuickTest window to suit your needs for each type of QuickTest session (view/edit, record, and run sessions). For example, you can display QuickTest in full view when creating or editing a test, and minimize the QuickTest window during a run session. For more information, see "Customizing the QuickTest Window Layout" on page 1144.

When you customize or restore the QuickTest window layout, QuickTest applies the changes to all document types and session types.

For more information, see:

- "Moving Panes" on page 1136
- "Showing and Hiding Panes" on page 1141
- "Floating and Docking Toolbars" on page 1144
- "Restoring the Default Layout of the QuickTest Window" on page 1144
- "Customizing the QuickTest Window Layout" on page 1144

Moving Panes

You can move the QuickTest window panes to suit your own personal preferences. You can rearrange the panes, and you can also change a pane to a tabbed pane, and vice versa.

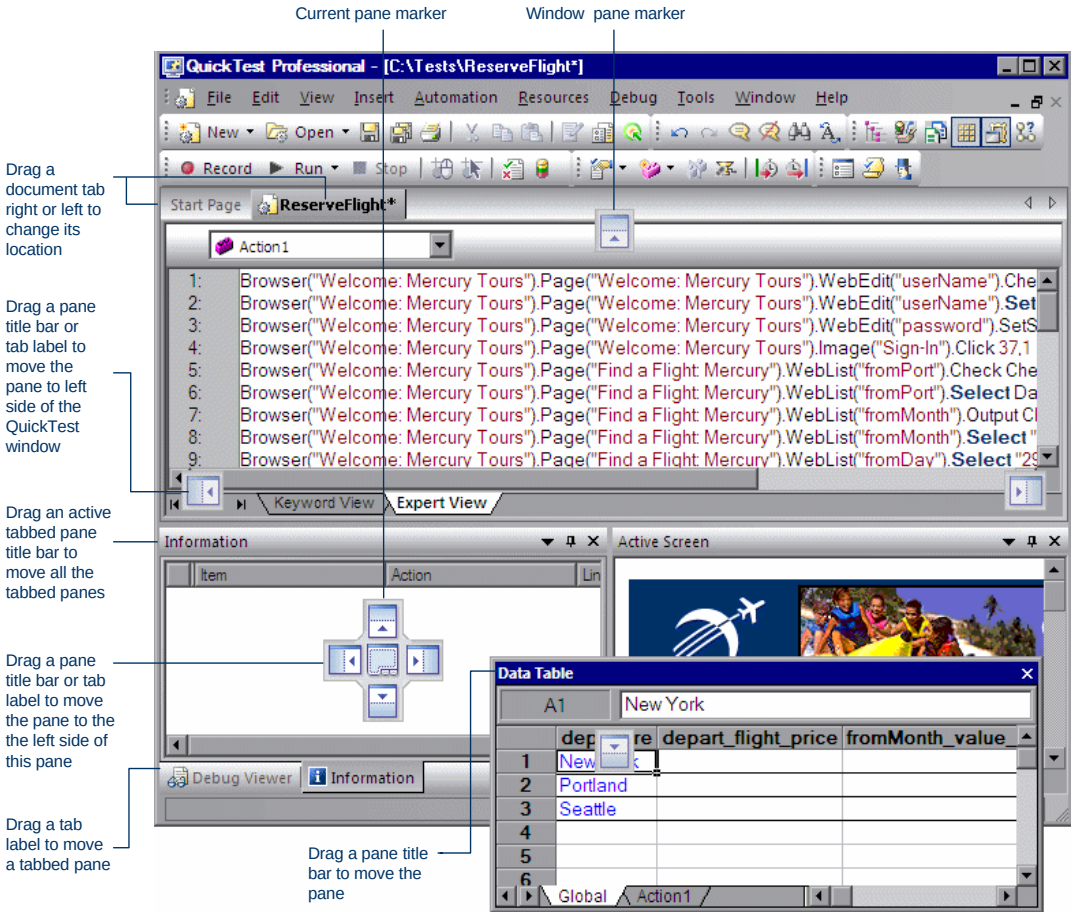
While dragging a pane, markers are displayed on the QuickTest window. If you hold the cursor over one of these markers, the area represented by the marker is shaded, enabling you to preview the window layout if the pane is moved to the selected position.

Tip: To move a dockable pane without snapping it into place, press CTRL while dragging it to the required location.

To move panes:

1 In the QuickTest window, drag the title bar or tab of the pane you want to move. (If the required pane is not displayed in the QuickTest window, you can select it from the **View** menu.)

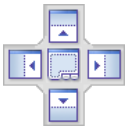
For example, you can move the Data Table tabbed pane located at the bottom left to be a new pane at the top right of the window. As you drag the pane, markers are displayed in the active pane and on each edge of the QuickTest window.



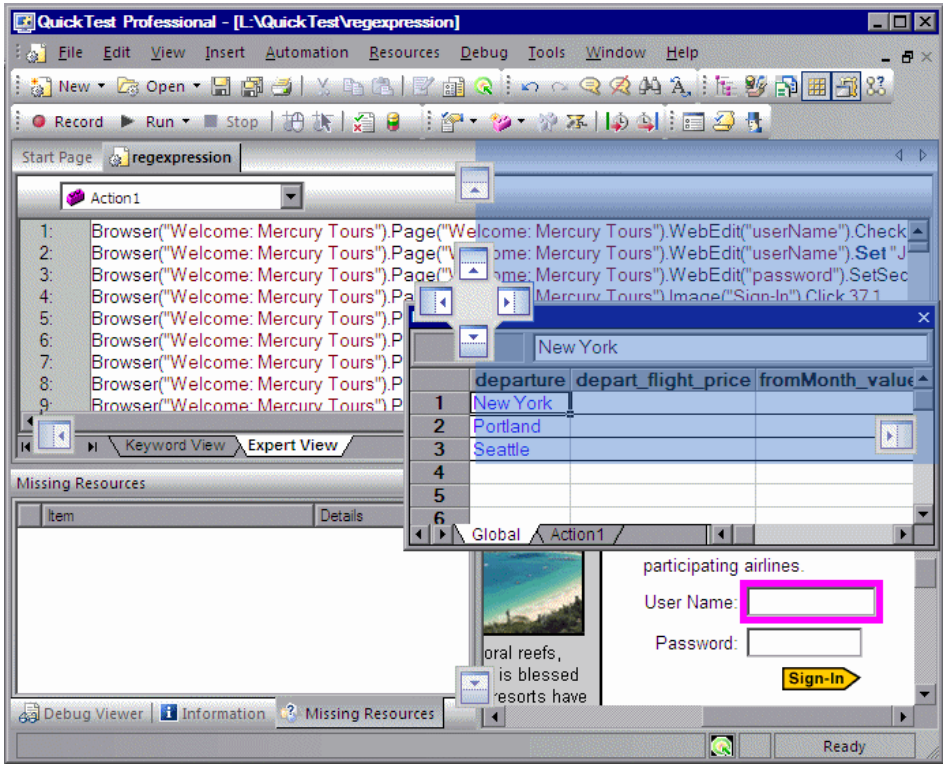
Tips:

- To move a single tabbed pane, drag the tab label. After you start dragging the tabbed pane, the tab is removed, and its title bar is displayed.
- To move all the tabbed panes, drag the title bar of the active tabbed pane.

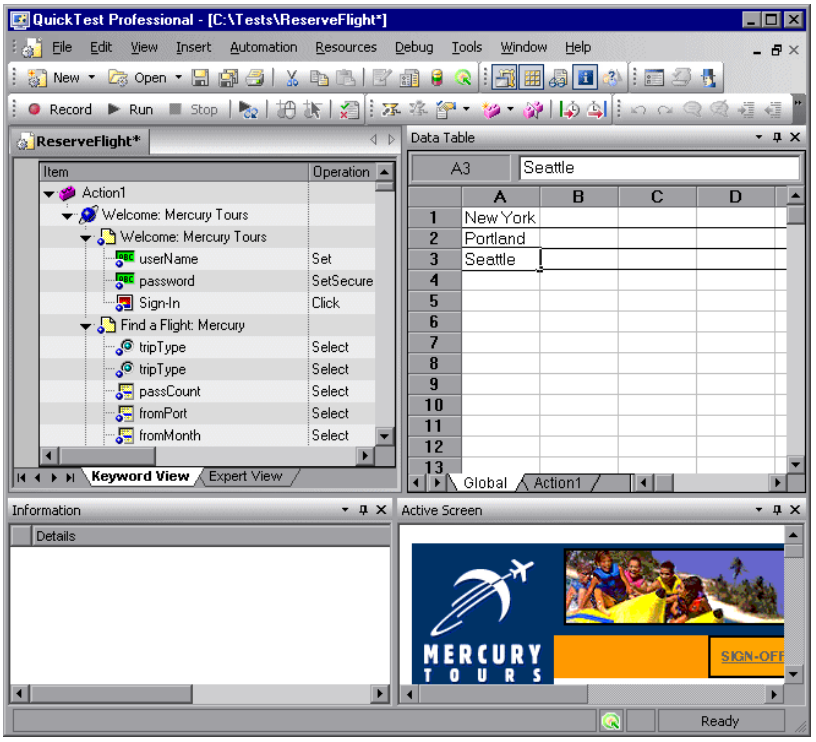
The following markers are displayed:

Type	Marker	Description
Current pane markers		Enables you to: ➤ position the pane as a new pane in the top, bottom, left or right half, or middle of the active pane, according to the arrow marker selected when you release the mouse button. ➤ position the pane as a new tabbed pane in the active window, by releasing the mouse button while selecting the center marker. **Note:** The center marker is displayed only if you are dragging a pane onto an existing pane (other than the document pane).
Window pane markers		Enables you to position the pane across the top of the QuickTest window.
		Enables you to position the pane across the right side of the QuickTest window.
		Enables you to position the pane across the bottom of the QuickTest window.
		Enables you to position the pane across the left side of the QuickTest window.

2 Drag the Data Table and hold the cursor over the active pane right-arrow marker, as shown below. A shaded area is displayed, indicating the new location of the pane, as shown below.



3 Release the mouse button. The Data Table snaps into place and is displayed as a new pane in the shaded area.



Tip: You can also leave the pane as a floating pane anywhere on the QuickTest window, or on your screen. For more information on floating panes, see “Showing and Hiding Panes” on page 1141.

---

4 Repeat this procedure for each pane you want to move.

Showing and Hiding Panes

After you move the panes to their default positions, you can decide whether these panes should be displayed at all times, or whether you want to auto-hide them, and only display them as required.

Panes can have one of the following states—docked or floating:

➤ **Docked panes.** Docked panes are fixed in a set position relative to the rest of the application. For example, when you move a pane to a position indicated by a marker, the pane is docked in that position.

You can decide whether to continuously display the docked panes in the QuickTest window, or to auto-hide them. Auto-hiding means that the pane is displayed as a side-tab on the edge of the QuickTest window, and is displayed only when you hold the cursor over the tab. After you select a different pane or side-tab, the auto-hidden pane closes and is displayed as a side-tab.

Note: If you auto-hide the Information pane, it is automatically displayed when syntax errors are detected in a test script.

By default, auto-hidden panes open across the QuickTest window, according to their position in the QuickTest window. For example, if the docked pane was positioned on the right side of the QuickTest window, it is displayed as a side tab on the right edge of the QuickTest window, and is displayed across the right side of the QuickTest window when selected.

Tip: To auto-hide all the tabbed panes, select the title bar of the active tabbed pane, right-click and select **Auto Hide**. The tabbed panes are displayed as a group of side-tabs on the edge of the QuickTest window, and each pane is displayed only when you hold the cursor over that side-tab.

➤ **Floating panes.** Floating panes are displayed on top of all other windows. They can be dragged to any position on your screen, even outside the QuickTest window. Floating panes have their own title bars.

Note: You cannot auto-hide floating panes or individual tabbed panes.

To show or hide panes:

In the QuickTest window, select the pane you want to auto-hide, and display as a side-tab on one of the edges of the QuickTest window. The following buttons may be displayed on the title bar:

Button	Description
	The **Menu** button enables you to select any of the following: ➤ **Floating.** Opens the pane on top of all the other windows and panes, with its own title bar ➤ **Docking.** Docks the pane to the QuickTest window. ➤ **Auto-hide.** Displays the pane as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window. ➤ **Hide.** Closes the pane.
	The **Auto Hide** button hides the pane. The pane is displayed as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window. To display the pane, hold the cursor over the side-tab. The button toggles to the **Dock** button, shown below.

Button	Description
	The **Dock** button docks the pane to the QuickTest window. The pane returns the position it was in before it was hidden, and the button toggles to the **Auto Hide** button, shown above.
	The **Close** button closes the pane. The pane is removed from the QuickTest window. To reopen the pane, select it from the **View** menu. **Tip:** You can also close a pane by right-clicking and choosing **Hide** from the context menu.

Tips:

- To auto-hide all the tabbed panes, select the title bar of the active tabbed pane, right-click and select **Auto Hide**. The tabbed panes are displayed as a group of side-tabs on the edge of the QuickTest window, and each pane is displayed only when you hold the cursor over that side-tab.
- You can float a pane by right-clicking the title bar, and choosing **Floating** from the context menu. The pane opens on top of all the other windows and panes, with its own title bar. To dock the pane, double-click the title bar, or right-click the title bar and select **Docking**. The pane returns to its previous position in the QuickTest window.

Floating and Docking Toolbars

You can float a toolbar by moving your cursor over the toolbar handle on the left of the toolbar and then dragging the toolbar to the required position. The toolbar is displayed with a title bar.

You can double-click the title bar of the menu to dock the menu and return it to its previous position in the QuickTest window, or you can close it by clicking the **Close** button.

Restoring the Default Layout of the QuickTest Window

You can restore the default QuickTest window layout for all document types at any time.

To restore the default layout:

1. Select **Tools** > **Options** > **General** node. The Options dialog box is displayed.
2. In the General pane, click the **Restore Layout** button. The panes and toolbars of all document types are restored to their default size and location.

For more information on the Options dialog box, see Chapter 44, "Setting Global Testing Options."

Customizing the QuickTest Window Layout

QuickTest works in several different modes: view/edit, record, and run. You may want to modify the QuickTest layout to match the functionality of a mode. For example, when recording, it is often convenient to have QuickTest partially visible. This enables you to watch steps being added as you record your test without viewing the Active Screen. When running a test, it is often convenient to minimize QuickTest so that you can view your application during the test run. When viewing or editing a test, it may be convenient to maximize the QuickTest window, with all panes showing.

QuickTest remembers the size and location of its main window and all of its panes for each mode. When QuickTest enters a mode, the layout reverts to the most recently used layout for that mode. This means that the main QuickTest window and each of its panes are maximized, minimized, or resized, based on the previous layout of the current mode.

To set the QuickTest layout for each mode:

1 Set the record mode:

a Open a new or existing test.

b Start a recording session.

c Record one step.

d Set all of your layout preferences for the recording mode.

e Stop the recording session.

2 Set the run mode:

a Enter a breakpoint before the first step in the test. This enables you to arrange the layout during the run session. For information on how to set a breakpoint, see "Setting Breakpoints" on page 1079.

b Run your test.

c When QuickTest reaches the breakpoint, set all of your layout preferences for the run mode.

d Stop the run session.

3 Set all of your layout preferences for the view/edit mode.

The layouts for all of these modes are now set. QuickTest applies the relevant layout each time it enters one of these modes.

Customizing Toolbars and Menus

You can use the Customize dialog box to create user-defined menus and to customize the appearance of existing menus and toolbars.

This section includes:

Customization Mode Options

While the Customize dialog box is open, QuickTest is in customization mode. The following options are available in the context-sensitive menu when you right-click the menu bar or toolbar buttons in customization mode:

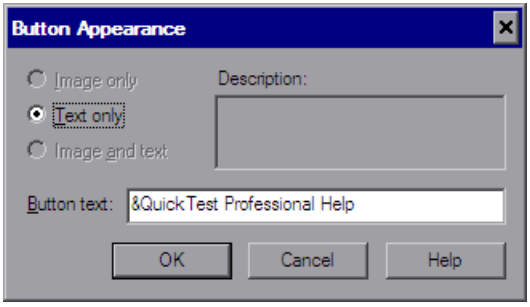
Option	Description
Restore Default	Restores the default setting for the button. This selection is disabled for menus.
Copy Button Image	Copies the button image to the clipboard. This selection is disabled for items that have no default image.

Option	Description
Delete	Deletes the menu or button. **To restore a button:** 1 Click the customize toolbar button  while in normal mode. 2 Select **Add or Remove Buttons.** 3 Select the menu whose button you want to restore and select **Restore Toolbar**. **To restore a menu from the menu bar:** Select the **Menu Bar** in the Toolbars tab and click the **Restore Selected** button. **Note:** Any customizations of the Menu bar will be lost. For more information, see "The Customize Dialog box - Toolbars Tab" on page 1152.
Button Appearance	Opens the Button Appearance dialog box. For more information, see "The Button Appearance Dialog Box" on page 1148.
Image	Displays the image for the button in the toolbar or menu. This selection is disabled for items that have no default image.
Text	Displays the text label for the button in the toolbar or menu. This selection is disabled for menus.
Image and Text	Displays the image and text label for the button or menu in the toolbar or menu. This selection is disabled for items that have no default image.
Start Group	Places a divider in the toolbar or menu before the current button to indicate a new group of buttons.

The Button Appearance Dialog Box

Description	Enables you to modify the appearance of a button or menu.
How to Access	Do one of the following: ➤ Select the **Tools** > **Customize** menu command, right-click a button or menu, and select **Button Appearance**. ➤ Click the customize toolbar button, select **Add or Remove Buttons** > **Customize,** then right-click a button or menu, and select **Button Appearance**. ➤ Right-click on the menu bar or any toolbar and select **Customize,** then right-click a button or menu, and select **Button Appearance**.

Below is an image of the Button Appearance dialog box:



Button Appearance Dialog Box Options

Option	Description
Image only	Displays the image for the button in the toolbar or menu. This radio button is disabled for items that have no default image.
Text only	Displays the text label for the button in the toolbar or menu.
Image and text	Displays the image and text label for the button or menu in the toolbar or menu. This radio button is disabled for items that have no default image.

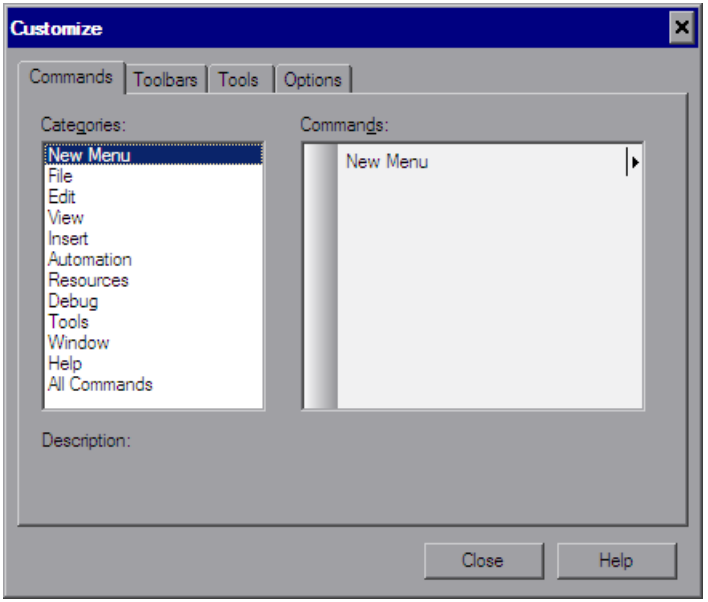
Option	Description
Description	The description of the button.
Button text	The text label for the button or menu. The text label for the button can be modified when either the **Text only** or **Image and text** radio buttons are selected. You can create a mnemonic (an underlined character for keyboard navigation) for any button text. Add the & character to the text label for the button before the letter you want to define as the mnemonic. Each button text can have only one mnemonic.

The Customize Dialog box - Commands Tab

Description	Enables you to customize toolbars and menus, and create new menus.
How to Access	Do one of the following: ➤ Select the **Tools** > **Customize** menu command and then click the **Commands** tab. ➤ Click the customize toolbar button  , select **Add or Remove Buttons** > **Customize**, and then click the **Commands** tab. ➤ Right-click on the menu bar or any toolbar and select **Customize** and then click the **Commands** tab.

Important Information	➤ See: ➤ "Customization Mode Options" on page 1146. ➤ "Considerations for Customizing Toolbars and Menus" on page 1158. **To add or remove default buttons from existing toolbars you can also:** 1 Right-click the customize toolbar button . 2 Select **Add or Remove Buttons.** 3 Select the menu whose buttons you want to modify. 4 Select or deselect the specific button. Toolbars are listed in the **Add or Remove Buttons** selection per row.
Learn More	**Primary task**: "To add a command to a toolbar or menu:" on page 1151.

Below is an image of the Customize Dialog box - Commands Tab:



Customize Dialog box - Commands Tab Dialog Box Options

Section	Description
Categories	A list of all of the menu items in the menu bar, with the addition of **New Menu** and **All Commands**.
Commands	A list of all the commands available in the menu item selected in the **Categories** list. Commands that appear in drop-down lists or sub-menus are listed as individual commands in the **Commands** section. For example, **Test** in the **New** drop-down list in the Standard toolbar is listed as the individual command **New : Test** in the **File** category.
Description	A description of the selected command in the **Command** list.

To add a command to a toolbar or menu:

1 Select **Tools** > **Customize Toolbars and Menus** and click the **Commands** tab.

2 In the **Categories** list, find and select the menu name that contains the command you want to add to the toolbar. To view all the available commands in alphabetical order select **All Commands**.

3 In the **Commands** list, select the command you want to add and drag it to a toolbar or the menu bar.

4 When you place the command over the menu bar or one of the toolbars, a marker is displayed indicating the location where the command will be placed. Drag the marker to the location where you want to add the command, and release the mouse button.

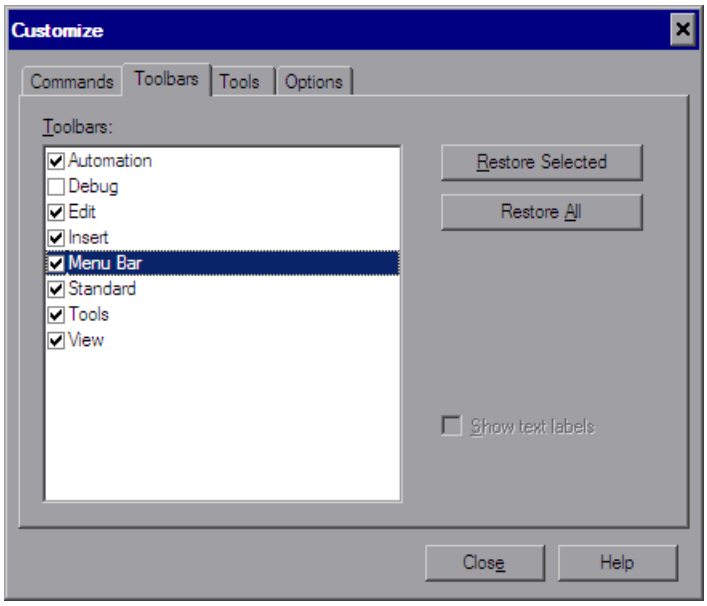
5 If you want to create a new menu select **New Menu** in the **Categories** list and drag the **New Menu** item to the menu bar or a toolbar. To create a name for the new menu see "The Button Appearance Dialog Box" on page 1148.

6 If you want to add a command to an existing menu, drag the command over the menu item. The menu item expands. Drag the marker to the location in the menu where you want to add the command, and release the mouse button.

The Customize Dialog box - Toolbars Tab

Description	Enables you to: ➤ show or hide toolbars or the menu bar. ➤ restore the default setting for one or all toolbars or the menu bar. ➤ display text labels for toolbar buttons.
How to Access	Do one of the following: ➤ Select the **Tools** > **Customize** menu command and then click the **Toolbars** tab. ➤ Click the customize toolbar button  , select **Add or Remove Buttons** > **Customize**, and then click the **Toolbars** tab. ➤ Right-click on the menu bar or any toolbar and select **Customize** and then click the **Toolbars** tab.
Important Information	➤ You can also show or hide toolbars using the **View** > **Toolbars** menu option or by right-clicking the toolbars area and selecting or deselecting a toolbar from the context menu. ➤ See: ➤ "Customization Mode Options" on page 1146. ➤ "Considerations for Customizing Toolbars and Menus" on page 1158.

Below is an image of the Customize Dialog box - Toolbars Tab:



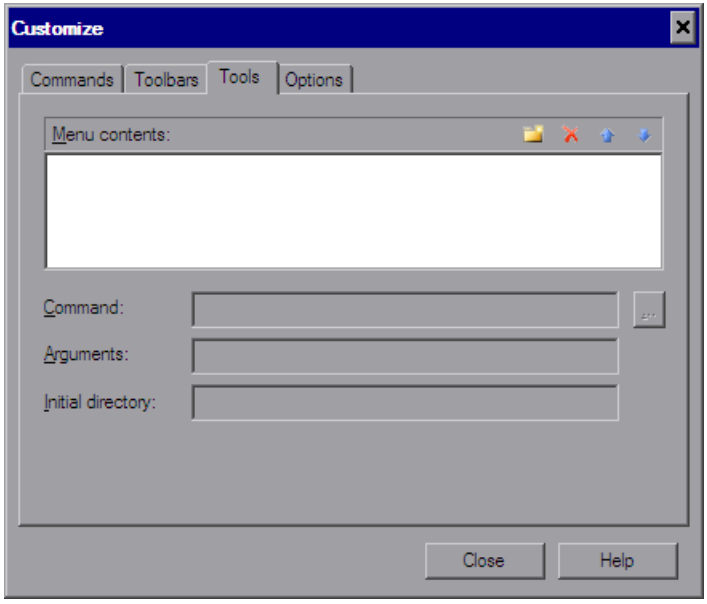
Customize Dialog box - Toolbars Tab Options

Option	Description
Toolbars	A list of the toolbars in the QuickTest window, with the addition of **Menu Bar**. Select or deselect a check box to show or hide a toolbar.
Restore Selected	Restores the default layout for the selected toolbar or the menu bar. **To restore the default layout for a toolbar you can also:** 1 Right-click the customize toolbar button . This is not available for the menu bar. 2 Select **Add or Remove Buttons.** 3 Select the toolbar whose layout you want to restore. 4 Select **Restore Toolbar.** Toolbars are listed in the **Add or Remove Buttons** selection per row of toolbars.
Restore All	Restores the default layout for all toolbars.
Show text labels	Displays text labels for the buttons in the currently highlighted toolbar. For buttons that have a text label by default (for example, the **Run** button), clearing this check box restores the default display, and the text labels are still displayed. To turn off text labels for a toolbar, highlight the toolbar in the **Toolbars** area and deselect the check box. This check box is disabled for the menu bar toolbar.

The Customize Dialog box - Tools Tab

Description	Enables you to add an item to the Tools menu so you can launch an application from the QuickTest menu.
How to Access	Do one of the following: ➤ Select the **Tools** > **Customize** menu command and then click the **Tools** tab. ➤ Click the customize toolbar button, select **Add or Remove Buttons** > **Customize**, and then click the **Tools** tab. ➤ Right-click on the menu bar or any toolbar and select **Customize** and then click the **Tools** tab.
Important Information	See: ➤ "Customization Mode Options" on page 1146. ➤ "Considerations for Customizing Toolbars and Menus" on page 1158.

Below is an image of the Customize Dialog box - Tools Tab:



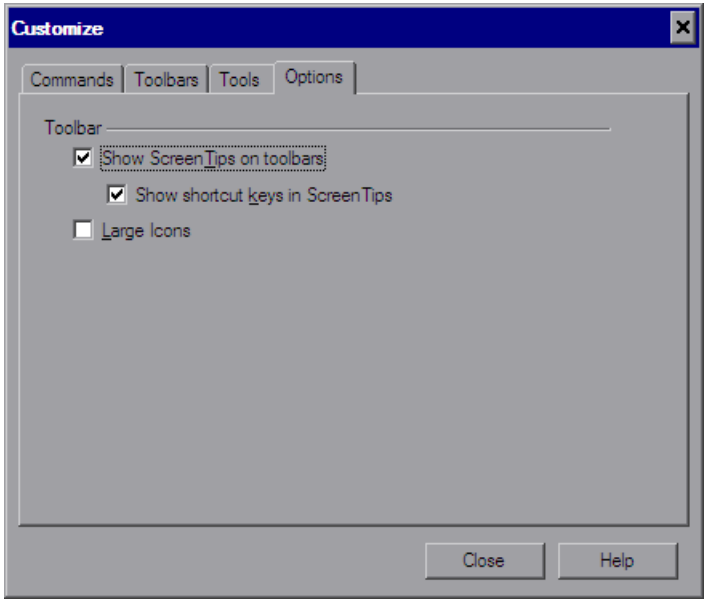
Customize Dialog box - Tools Tab Options

Option	Description
Menu Contents:	A list of the items added to the tools menu.
	**New**. Enables you to add a new item to the **Tools** menu. A blank line is added to the **Menu Contents** area. Enter a name for the new item.
	**Delete**. Enables you to delete the item selected in the **Menu Contents** list from the **Tools** menu.
	**Move Item Up**. Enables you to move the selected item up in the **Tools** menu.
	**Move Item Down**. Enables you to move the selected item down in the **Tools** menu.
Command:	The application for which you want to add an item to the Tools menu. Click the browse button  and navigate to the application you want to add. **Note:** The **Command** box should contain only the file name and path for the application. If you want to add command line arguments, use the **Arguments** box. **Tip:** You can specify a document or other file associated with an application in the file system, for example, c:\tmp\a.txt. In this case, QuickTest automatically opens the specified file in the associated application (Notepad in this example). If you use this option, QuickTest ignores any defined program arguments.
Arguments:	**Optional.** Instructs QuickTest to open the application using the specified command line arguments.
Initial directory:	**Optional**. Specifies the current working folder for the application. The initial directory is used by the application to search for related files. If an initial directory is not specified, the executable folder is used as the initial directory.

The Customize Dialog box - Options Tab

Description	Enables you to display ScreenTips (tooltips), shortcut keys and large or small icons in the QuickTest display.
How to Access	Do one of the following: ➤ Select the **Tools** > **Customize** menu command and then click the **Options** tab. ➤ Click the customize toolbar button, select **Add or Remove Buttons** > **Customize**, and then click the **Options** tab. ➤ Right-click on the menu bar or any toolbar and select **Customize** and then click the **Options** tab.
Important Information	See: ➤ "Customization Mode Options" on page 1146. ➤ "Considerations for Customizing Toolbars and Menus" on page 1158.

Below is an image of the Customize Dialog box - Options Tab:



Customize Dialog box - Options Tab Options

Option	Description
Show ScreenTips on toolbars	Turns ScreenTips (tooltips) on or off. Select this check box to display ScreenTips in the QuickTest display. ➤ **Show shortcut keys in ScreenTips**. Select this check box to display shortcut keys in the ScreenTips.
Large Icons	Turns large icons on or off. Select this check box to display large icons in the QuickTest display.

Considerations for Customizing Toolbars and Menus

- Toolbar and menu customization settings are created and saved for each Windows user.
- You can delete any button or command while the Customize Dialog box is open. Drag the toolbar button you want to remove from the toolbar to any location outside the toolbars area. The toolbar button is removed.
- You can restore the default buttons and layout for a selected toolbar or for all toolbars using the **Restore** or **Restore All** buttons in the Toolbars tab. You can also restore the default buttons and layout for a toolbar by right-clicking the customize toolbar button  , selecting **Add or Remove Buttons**, selecting the toolbar whose settings you want to restore, and then selecting **Restore Toolbar**.
- While the Customize Dialog box is open you can drag toolbar buttons from one toolbar to another toolbar and drag and drop to change the order of items in a menu.
- Some QuickTest add-ins add commands or menus to the QuickTest window. If you are working with add-ins and customize the toolbars, consider the following:
  - QuickTest will remember your customizations as long as you continue working with those add-ins, even if you close and reopen QuickTest.
  - When QuickTest is run without those add-ins, all commands and menus added by the add-ins are removed from the QuickTest window.

➤ If you customize the toolbars first and then run QuickTest with add-ins, the additional commands and menus will be placed as close as possible to their intended locations, based on adjacent items.

Working with Multiple Documents

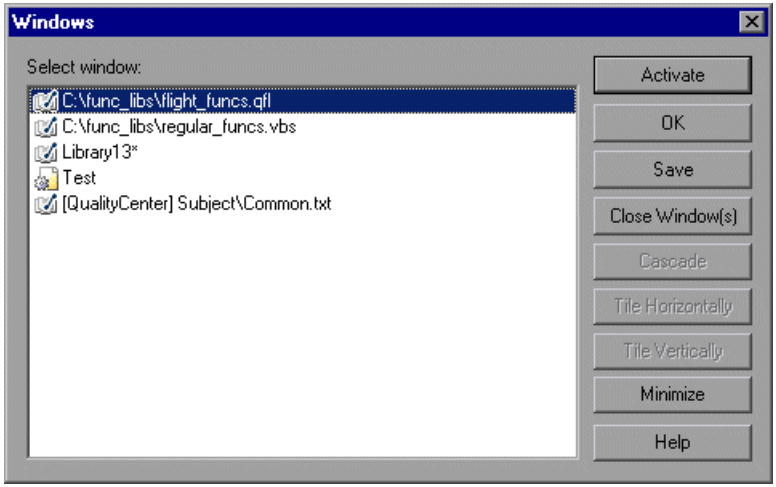
QuickTest enables you to open and work on one test at a time. In addition, you can open and work on multiple function libraries simultaneously. You can open any function library, regardless of whether it is associated with the currently open test.

The **Windows** menu options enable you to locate and activate (bring into focus) an open document window, select how the open document windows are arranged in the QuickTest window, or close all the open function library windows.

You can also use the Windows dialog box to manage your open QuickTest document windows.

To work with multiple documents using the Windows dialog box:

1 Select **Window > Windows**. The Windows dialog box opens.



The Windows dialog box displays a list of the open document windows, including the open test, as well as all the currently open function library windows.

2 The Windows dialog box contains the following buttons, enabling you to manage your open documents:

Button	Description
Activate	Brings the selected document into focus in the QuickTest window.
OK	Closes the Windows dialog box.
Save	Saves the selected documents.
Close Window(s)	Closes the selected function libraries.
Cascade	Arranges the selected documents in a cascading order that overlaps.
Tile Horizontally	Arranges the selected documents side-by-side horizontally, without overlapping.
Tile Vertically	Arranges the selected documents side-by-side vertically, without overlapping.
Minimize	Minimizes the selected documents.
Help	Displays the QuickTest Professional Help topic for this dialog box.

3 Click **OK** to close the Windows dialog box.

38

Managing Resources

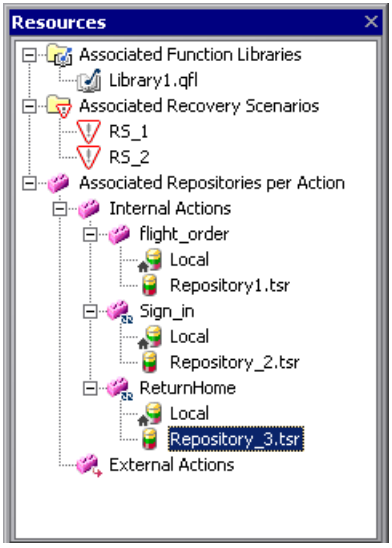
QuickTest enables you to manage the resources associated with your test in one pane. Using the Resources pane, you can associate, remove, open, change the priority, and otherwise manage the function libraries, recovery scenarios, and object repositories in your test.

This chapter includes:

The Resources Pane

Description	Enables you to add, remove, and manage, view, and open most associated resources for your test.
How to Access	Do one of the following: ➤ Select the **View** > **Resources** menu option. ➤ Click the **Resources Pane** toolbar button  .
Important Information	Tests and actions are associated with resources, such as function libraries, recovery scenarios, and object repositories. The resources in the Resources pane are displayed for the current test. Function libraries and recovery scenarios are grouped by resource type. Object repositories are grouped by action. The resources in the Resources pane are displayed in a tree hierarchy. Right-clicking a node in the tree opens the context menu for that resource. Some options are accessible through the context menu of the root node for a resource and some options are accessible through the context menu of the specific resource.

Below is an image of the Resources pane:



Resources Pane Tree Node Types

Node Type	Description
Associated Function Library	Lists all of the function libraries currently associated with your test. **Root node context menu option:** **Associate Function Library.** Opens the Open Function Library dialog box, enabling you to associate a function library with your test. **Function library node context menu options:** ➤ **Open Function Library.** Opens the selected function library in the QuickTest Function Library window. You can also double-click to open a function library. ➤ **Remove Function Library from List.** Removes the selected function library from your test. ➤ **Move Up** or **Move Down**. Moves the selected function library up or down the priority list of associated function libraries. **See also:** "Working with User-Defined Functions and Function Libraries" on page 905

Node Type	Description
Associated Recovery Scenarios	Lists all of the recovery scenarios currently associated with your test. **Root node context menu option:** **Associate Recovery Scenario**. Opens the Add Recovery Scenario dialog box. For information, see "Adding Recovery Scenarios to Your Test" on page 1373. **Recovery scenario node context menu options:** ➤ **Recovery Scenario Properties**. Opens the Recovery Scenario Properties dialog box. This enables you to view properties for the recovery scenario in read-only, such as trigger events and recovery operations. You can also double-click a recovery scenario to open the Recovery Scenario Properties dialog box. For information, see "Viewing Recovery Scenario Properties" on page 1368. ➤ **Remove Recovery Scenario from List**. Removes the selected recovery scenario from your test. ➤ **Move Up** or **Move Down**. Moves the selected recovery scenario up or down the priority list of associated recovery scenarios. ➤ **Disable Recovery Scenario** / **Enable Recovery Scenario**. Disables or enables the selected recovery scenario. **See also:** "Defining and Using Recovery Scenarios" on page 1329

Node Type	Description
Associated Repositories per Action	Lists all of the object repositories currently associated with all of the actions in your test. The nodes in this part of the Resources pane are organized according to the following hierarchy: **Associated Repositories per Action** node **Internal Actions** and **External Actions** nodes individual actions local and shared object repositories associated with the individual action (if any) The individual actions include all of the actions stored with your test, even when they are not called by your test. **Action node context menu options:** ➤ **Associate Repository with Action**. Opens the Open Shared Object Repository dialog box, enabling you to associate an object repository with the selected action. This option is disabled for external actions. ➤ **Action Properties**. Opens the Action Properties dialog box, enabling you to define options for the stored action. You can modify an action name, add or modify an action description, and set an action as reusable or non-reusable. For an external action, you can set the Data Table definitions. For more information, see "Setting Action Properties" on page 441. ➤ **Delete Action**. Removes the action from the test. **Repository node context menu options:** ➤ **Open Repository**. Opens the Object Repository Window-Local Object Repository for local object repositories and the Object Repository Manager for shared object repositories. Double-clicking an object repository also opens the relevant repository window. ➤ **Remove Repository from List**. Removes the selected object repository from the action. ➤ **Move Up** or **Move Down**. Modifies the priority of the selected object repository when you move it up or down in the list of associated repositories. **See also:** "Managing Test Objects in Object Repositories" on page 135

39

Adding Keywords to Your Test

QuickTest enables you to view and add the available keywords to your test in one pane.

This chapter includes:

Understanding the Available Keywords Pane

The Available Keywords pane displays the keywords available to your test. It enables you to view the available objects and calls to functions, and also enables you to drag and drop them into your test. When you drag and drop an object into your action, QuickTest inserts a step with the default operation for that object. When you drag and drop a function into your test, QuickTest inserts a call to that function.

For example, if you drag and drop a button object into your action, a step is added using the button with a **Click** operation (the default operation for a button object).

If you drag and drop a function into your test, a comment and call to that function is added. The comment indicates that a call to the function was added to your test and indicates any necessary arguments. You need to provide the arguments for that function to your test. In the Keyword view, a tooltip displays the required arguments for the function. In the Expert view, IntelliSense displays the required arguments for the function.

You can also drag and drop test objects from other locations. For more information, see:

- "The Object Repository Window" on page 183
- "Adding Test Objects to Your Test Using the Object Repository Manager" on page 225

The Available Keywords pane can display the keywords available to your test sorted by resource or sorted by keyword.

To view the Available Keywords pane:

Click the **Available Keywords Pane** button or select **View** > **Available Keywords**.

Keywords Sorted by Resource

You can display the keywords sorted by resource by clicking the **Sort by Resource** button. Keywords are grouped by their type (library functions, local functions, objects) and then by the specific resource for that type.

- Functions in each function library are sorted alphabetically.
- Objects in each object repository are grouped by the page or window in which they appear in the application, then by the object type. They are then sorted alphabetically.
- Right-clicking a keyword enables you to open the keyword's resource or copy the selected keyword to the clipboard.
- Double-clicking a keyword opens the keyword's resource and points to the selected keyword.

Keywords Sorted by Keyword

You can display the keywords sorted by keyword by clicking the **Sort by Keyword** button. Keywords are grouped by their type (library functions, local functions, objects) regardless of their resource.

- All available functions are sorted alphabetically.
- All available objects are grouped by the page or window in which they appear in the application, then by the object type. They are then sorted alphabetically.

Note: If two keywords have the same name, they are displayed according to the priority of their resources.

- Right-clicking a keyword enables you to open the keyword's resource or copy the selected keyword to the clipboard.
- Double-clicking a keyword opens the keyword's resource and points to the selected keyword.

40

Managing QuickTest Tasks and Comments

QuickTest enables you to create and manage tasks in your tests, and to create and manage TODO comments in your actions and function libraries.

This chapter includes:

Working with Tasks and TODO Comments

QuickTest enables you to create and manage tasks and TODO comments about issues that need to be handled in your tests and function libraries. For example, you can provide instructions to someone else during a handover, or remind yourself to do something.

Tasks are test-related reminders that are linked to the currently open test. You create and manage tasks using the To Do pane and the Task Editor dialog box.

TODO comments are reminders that are inserted as comment steps adjacent to the relevant steps in an action or function library. You can access TODO comments from the To Do pane or directly from the testing document.

If needed, you can export your tasks and TODO comments to Microsoft Excel or an XML file.

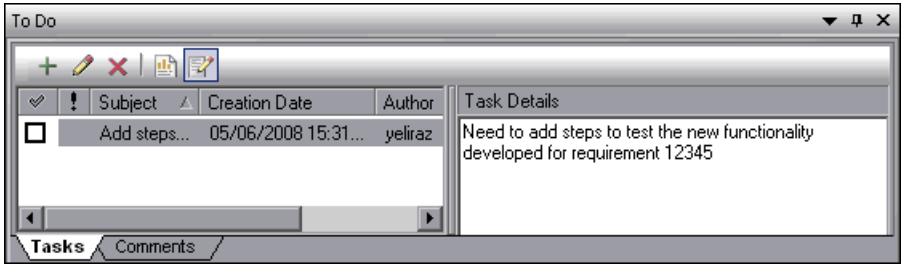
The To Do Pane

Description	Enables you to view and manage your test-related tasks and TODO comments. The To Do pane contains the following tabs: ➤ **Tasks tab.** Enables you to create and manage your test-related tasks. For more information, see "The To Do Pane: Tasks Tab" on page 1171. ➤ **Comments tab.** Enables you to view and access TODO comments in an action or currently open function library. For more information, see "The To Do Pane: Comments Tab" on page 1174.
How to Access	➤ Select the **View > To Do** menu item. ➤ Click the **To Do Pane** toolbar button  . **Note:** The To Do pane opens automatically when you open a test that contains tasks.
Learn More	**Conceptual overview**: "Working with Tasks and TODO Comments" on page 1169 **Additional related topics**: "The Task Editor Dialog Box" on page 1177

The To Do Pane: Tasks Tab

Description	The tasks tab displays all of the tasks defined for the currently open test. You define tasks using the Task Editor dialog box. Tasks are saved with the test.
How to Access	**View** menu > **To Do** item > **Tasks** tab
Learn More	**Conceptual overview**: "Working with Tasks and TODO Comments" on page 1169 **Additional related topics**: ➤ "The To Do Pane: Comments Tab" on page 1174 ➤ "The Task Editor Dialog Box" on page 1177

Below is an image of the Tasks tab in the To Do pane:



Tasks Tab Details

The Tasks tab displays all tasks that were created for this test using the Task Editor dialog box.

Tasks Tab Toolbar

	Toolbar Option	Shortcut Key	Description
	**Add Task**	INSERT	Opens the Task Editor dialog box (described on page 1177), enabling you to add a new task to the Tasks tab in the To Do pane.
	**Edit Task**	ENTER	Opens the Task Editor dialog box (described on page 1177), enabling you to modify the selected task or mark it as complete.
	**Delete Task**	DELETE	Removes the selected task from the To Do pane.
	**Export TODO List**	**N/A**	Saves the tasks to an external file, such as a text file. You can save the tasks in the list in any of the following formats: ➤ XML (Extensible Markup Language) ➤ XLS (Microsoft Excel file) ➤ CSV (Comma-Separated Values file)
	**Show/Hide Task Details**	**N/A**	Opens or closes the Task Details pane on the right side of the To Do pane, displaying more information about a selected task.

Tasks Tab Columns

You can click on a column header to sort by that column.

Column	Description
Completed	Indicates whether the task was fully implemented. When you mark a task as complete, a strike-through format is applied to the task in the pane, indicating that the task is completed. Possible values: ☑ **Task completed** ☐ **Task not completed** **Tip:** You can also mark a task as complete by selecting the **Task completed** check box in the Task Editor dialog box.
Priority	Indicates the importance of the task. Possible values: ➤ **High Priority** ➤ **Normal Priority** ➤ **Low Priority**
Subject	Indicates the topic of the task.
Creation Date	Indicates the date and time that the Task Editor was opened to create the current task.
Author	Indicates the name of the user who created the task.
Assigned To	Indicates the name of the user who is responsible for handling the task.
Task Details	When enabled, displays a textual description of the task.

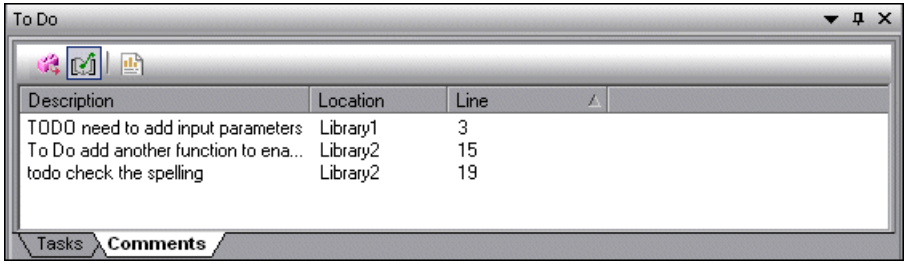
Tasks Tab Context Menu Options

Context Menu Option	Description
Sort By	Enables you to choose a column by which to sort the tasks in the tab.
Duplicate	Creates a copy of the selected task and inserts it in the Tasks tab. This is useful if you want to create a new task that is similar to an existing one.
Delete Task	Permanently removes the task from the Tasks tab in the To Do pane.

The To Do Pane: Comments Tab

Description	The Comments tab can display any comment step that begins with any of the following permutations of the words **to do**: To Do, todo, to-do, or TODO (not case-sensitive) **Example:** 'to DO need to ask Sarah to add design steps **Note:** The text displayed in the Comments tab is limited to 260 characters. If the text exceeds this limit, and you want to view the entire comment, you can jump to the comment in the testing document by double-clicking the comment line in the Comments tab. You can show TODO comments in: ➤ The current test's local actions. ➤ Any external actions associated with the test. ➤ Any open function library.
How to Access	**View** menu > **To Do** item > **Comments** tab
Learn More	**Conceptual overview**: "Working with Tasks and TODO Comments" on page 1169 **Additional related topics**: ➤ "The To Do Pane: Tasks Tab" on page 1171 ➤ "The Task Editor Dialog Box" on page 1177

Below is an image of the Comments tab in the To Do pane:



Comments Tab Details

By default, the Comments tab displays all TODO comment steps in the test's local actions. You can also choose to view TODO comment steps that are located in external actions called by the test and in currently open function libraries.

Comments Tab Toolbar

	Toolbar Option	Description
	Comments in External Actions	Toggle button that enables you to display or hide any TODO comments from external actions.
	Comments in Open Function Libraries	Toggle button that enables you to display or hide any TODO comments from currently open function libraries (in addition to the TODO comments from the local actions).
	Export TODO List	Saves the TODO comments to an external file, such as a text file. You can save the list of TODO comments in any of the following formats: ➤ XML (Extensible Markup Language) ➤ XLS (Microsoft Excel file) ➤ CSV (Comma-Separated Values file)

Comments Tab Columns

Column	Description
Description	Displays the text of the TODO comment.
Location	Specifies the name of the action or the path of the function library containing the TODO comment.
Line	Specifies the line number of the TODO comment in the action or function library.

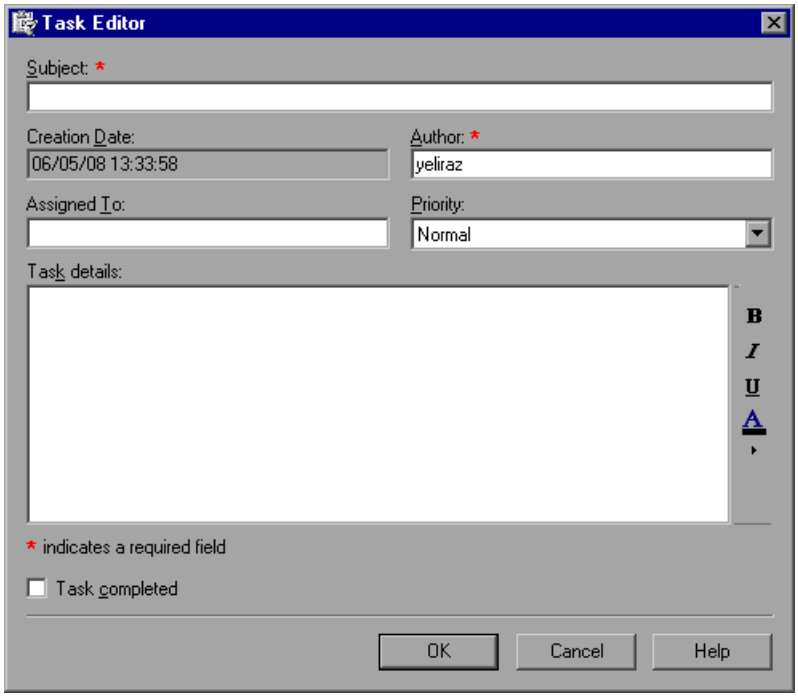
Comments Tab Context Menu Options

Context Menu Option	Description
Sort By	Enables you to choose a column by which to sort the TODO comments in the tab.
Go To Comment Line	Moves the cursor to the comment line in the action or function library. **Tip:** You can also double-click a comment in the Comments tab or press ENTER to move the cursor to the comment line in the action or function library.

The Task Editor Dialog Box

Description	Enables you to add a task to the To Do pane, edit an existing task, or to mark a task as complete.
How to Access	(Accessed from the Tasks tab in the To Do pane: **View** menu > **To Do** item > **Tasks** tab) In the Tasks tab, do one of the following: ➤ Click the **Add Task** button or press INSERT. ➤ Select a task and click the **Edit Task** button or press ENTER.
Learn More	**Conceptual overview**: "Working with Tasks and TODO Comments" on page 1169 **Additional related topics**: "The To Do Pane" on page 1170

Below is an image of the Task Editor dialog box:



Task Editor Dialog Box Options

You create and edit tasks using the Task Editor dialog box. Fields marked with a red asterisk (*) are mandatory.

Option	Description
Subject	A descriptive topic name for the task. You can enter up to 260 characters. (Mandatory field)
Creation Date	The date and time that the Task Editor was opened to create the current task. (Read-only)
Author	The automatically generated name of the user who created the task. You can modify the **Author** field when creating a task but not when modifying it. (Mandatory field upon creation)
Assigned To	The name of the user who is responsible for handling the task.
Priority	The importance of the task. Possible values: ➤ **High Priority** ➤ **Normal Priority** ➤ **Low Priority**
Completed	Indicates whether the task was fully implemented. Possible values: ☑ **Task completed** ☐ **Task not completed**
Task Details	A textual description of the task. You can modify the font style (bold, italic, and underline) and color to highlight various parts of the task details.

41

Handling Missing Resources

If a test has resources that cannot be found, such as missing shared object repositories or calls to missing actions, or if it uses a repository parameter that does not have a defined value, QuickTest indicates this in the Missing Resources pane. If one of the resources listed in this pane is unavailable during a run session, the test may fail. You can map a missing resource, or you can remove it from the test, as required.

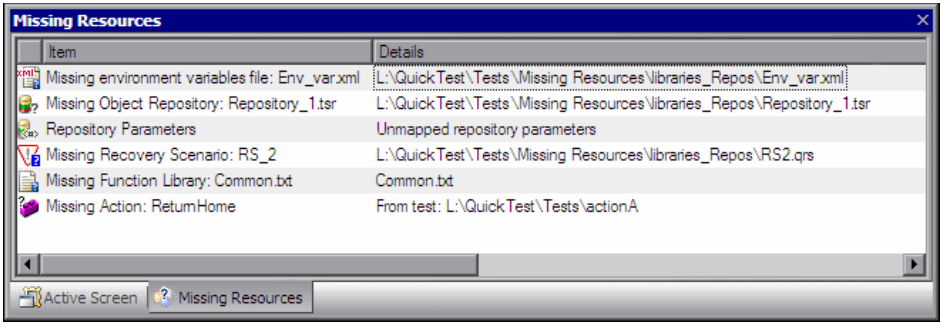
This chapter includes:

About Handling Missing Resources

Each time you open a test, QuickTest verifies that the resources specified for the test are available.

If one or more resources cannot be found, QuickTest opens the Missing Resources pane, if the pane is not already open. The Missing Resources pane provides a list of all resources that are currently unavailable, along with the location where QuickTest expected to find the resource, when available. The Missing Resources pane then enables you to locate or remove them from your test.

After you successfully handle a missing resource, QuickTest removes it from the pane.



The Missing Resources pane may list any of the following types of missing resources:

- **Missing action.** If a test contains an action that cannot be found, QuickTest specifies the path it uses to search for the test containing the missing action. For more information, see “Handling Missing Actions” on page 1183.
- **Missing environment variable file**. If a test loads user-defined environment variables from an external file that cannot be found, QuickTest specifies the path it uses to search for the missing XML file. For more information see, “Handling Missing Environment Variables Files” on page 1188.
- **Missing function library**. If a test is associated with a function library that cannot be found, QuickTest specifies the path it uses to search for the missing function library. For more information see, “Handling Missing Function Libraries” on page 1189.

➤ **Missing object repository.** If a test is associated with a shared object repository that cannot be found, QuickTest specifies the path it uses to search for the missing object repository. For more information, see "Handling Missing Shared Object Repositories" on page 1191.

➤ **Missing recovery scenario**. If a test is associated with a recovery scenario that cannot be found, QuickTest specifies the path it uses to search for the missing recovery scenario. For more information see, "Handling Missing Recovery Scenarios" on page 1192.

➤ **Repository parameters.** If a test has at least one test object with a property value that is parameterized using a repository parameter that does not have a default value, QuickTest adds this generic item to the Missing Resources pane. For more information, see "Handling Unmapped Shared Object Repository Parameter Values" on page 1194.

Note: In the various screens where a missing resource is used (for example, the Keyword View and test settings) QuickTest indicates that a resource is missing with a special icon or text.

Filtering the Missing Resources Pane

You can choose to display all missing resources in the Missing Resources pane, or only one type of missing resource.

To filter the list of displayed missing resources:

Right-click in the Missing Resources pane and select one of the following:

➤ **All.** Displays a list of all missing resources in your test.

➤ **Actions.** Displays a row for each missing action, specifying the path QuickTest uses to search for each test that contains a missing action.

➤ **Environment Variable File.** Displays the external XML file QuickTest uses to store user-defined environment variables.

➤ **Function Libraries.** Displays a row for each function library that cannot be found, specifying the path QuickTest uses to search for the function library.

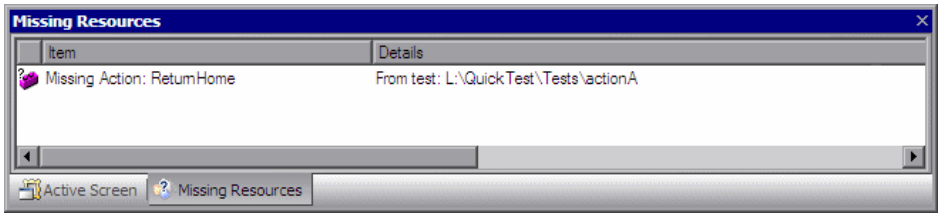
- **Object Repositories.** Displays a row for each shared object repository that cannot be found, specifying the path QuickTest uses to search for the shared object repository.
- **Recovery Scenarios.** Displays a row for each recovery scenario that cannot be found, specifying the path QuickTest uses to search for the recovery scenario.
- **Repository Parameters.** Displays a generic row indicating that at least one test object in the repository has at least one property value that uses a repository parameter that does not have a default value.

The Missing Resources pane is filtered according to the selected resource type and the following indication of the applied filter is shown at the bottom of the pane:

You can cancel the filter and show all missing resources again by clicking the ✕ icon on the left of the filter indication.

Handling Missing Actions

If your test contains a call to one or more actions that cannot be found, QuickTest lists these actions in the Missing Resources pane.



In addition, if the Test Flow contains a call to a particular action that is contained in the test, but the action cannot be found, QuickTest lists the action in the Missing Resources pane. For example, suppose that when you created a test, you inserted a call to a new, reusable action. Later, you deleted the call to that action by choosing the **Delete the selected call to the action** in the Delete Action dialog box (described on page 460). The action is still referenced by the test even though you deleted the call to it, and QuickTest will list it in the Missing Resources pane if it cannot be found.

The Missing Resources pane enables you to resolve a missing action by:

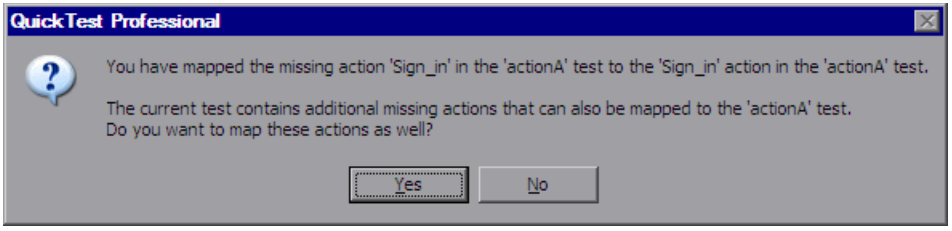
- Locating Missing Actions
- Removing Missing Actions

Note: If a test is opened in read-only format, you cannot view or map its missing actions.

Locating Missing Actions

The Missing Resources pane enables you to locate missing actions in your test.

If your test contains calls to more than one missing action, when you locate the missing action in another test, QuickTest may identify additional missing actions that are found in the same test, as shown in the following example:

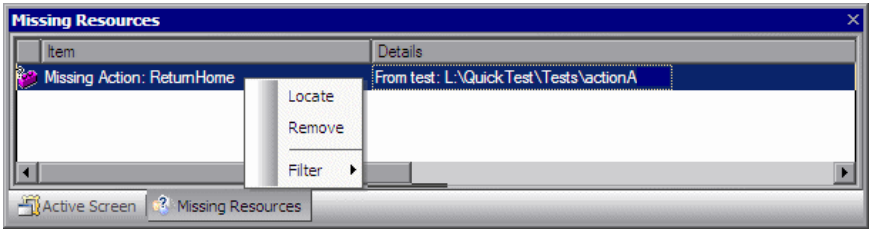


This can occur, for example, if the source test, which contains the actions that are being called was renamed or was moved to another folder.

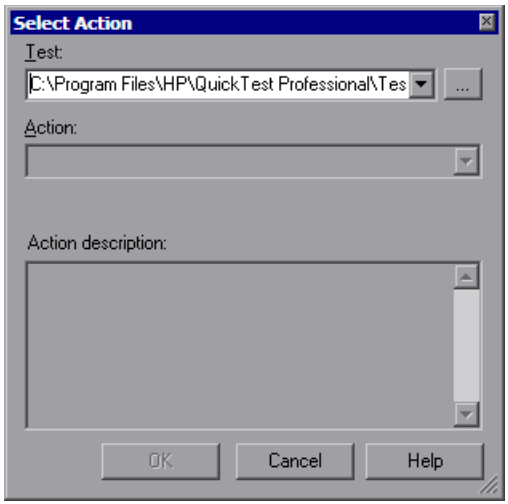
You can instruct QuickTest to locate these actions simultaneously, or you can handle each call to a missing action individually.

To locate a missing action:

1 In the Missing Resources pane, right-click the action you want to locate and select **Locate** from the context-sensitive menu or double-click the action you want to locate.



The Select Action dialog box opens.



When the Select Action dialog box opens, the **Test** box displays either the name of the test containing the missing action (if QuickTest can identify the source test), or **<Current Test>**.

Note: If the missing action is a nested action that is called from another test, you cannot use the **Locate** button to browse to that action. Instead, you must resolve the missing action from within the external test. For example, if ActionAA (in TestA) calls ActionBB (from TestB), and ActionBB calls ActionCC (from TestC), if you open TestA and the call to ActionCC is missing, then you can only resolve the missing action by opening TestB and locating ActionCC. (You cannot resolve it from within TestA.)

2 Click the browse button to find the test that contains the action you want to locate. The **Action** box displays all reusable actions in the test you select.

Notes:

- When you select a test, the **Test** box is renamed to **From test**. If the test you select contains reusable actions, these are listed in the **Action** box.
- You can enter a Quality Center folder or a relative path in the **Test/From test** box. If you enter a relative path, QuickTest searches for the test in the folders listed in the Folders pane of the Options dialog box. For more information, see "Setting Folder Testing Options" on page 1237 and "Using Relative Paths in QuickTest" on page 316.
  If you are working with the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.

---

3 In the **Action** list, select the action you want to call. When you select an action, its type (Reusable Action) and description, if one exists, are displayed. This helps you identify the action you want to call. For more information on action descriptions, see "Setting General Action Properties" on page 443.

4 Click **OK**. QuickTest updates the test with your changes and removes the action from the Missing Resources pane.

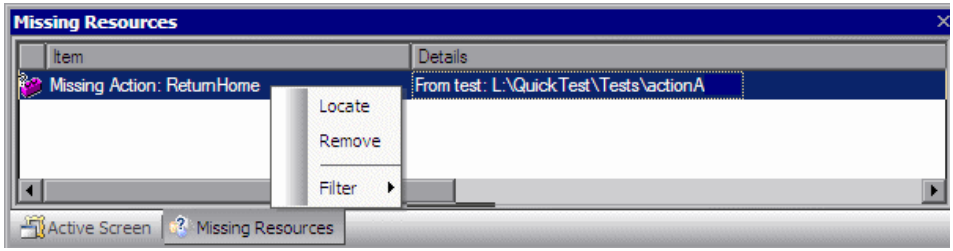
Note: If your test contains additional missing actions that can be located in the same test, QuickTest opens a message box asking you if you want to map these actions as well. Click **Yes** to map all relevant actions, or click **No** to map only the action you specified.

Removing Missing Actions

You can remove a missing action from a test if it is not needed.

To remove a missing action:

In the Missing Resources pane, right-click the action you want to remove and select **Remove** from the context-sensitive menu.

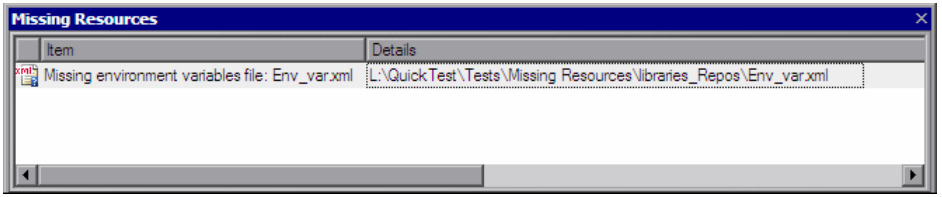


A confirmation message is displayed. Click **OK** to remove the missing action. QuickTest removes the action from your test and removes the action from the Missing Resources pane.

Note: If your test contains additional missing actions in the same test, QuickTest opens a message box asking whether you want to remove all the actions with the same path. Click **Yes** to remove all the missing actions in the same path, or click **No** to remove only the action you specified.

Handling Missing Environment Variables Files

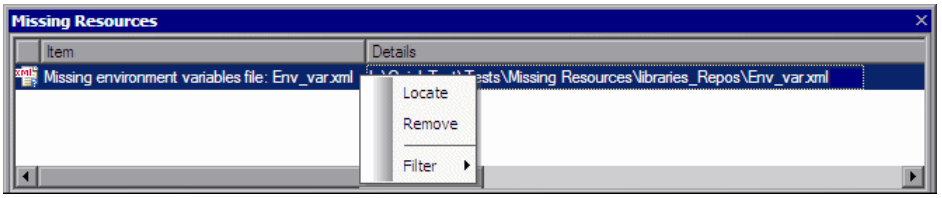
When you open a test that uses an external environment variables file, QuickTest verifies that the file is accessible. If an external environment variables file cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your test.



The Missing Resources pane enables you to resolve a missing external environment variables file by locating or removing it.

To locate a missing external environment variables file:

1 Right-click the missing environment variable file you want to locate and select **Locate** from the context-sensitive menu or double-click the missing environment variable file you want to locate.

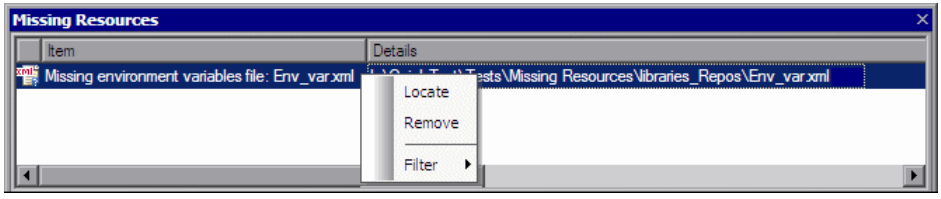


The Locate Environment Variable File dialog box opens.

2 Browse to the environment variable file you want to use with your test and click **Open**. The selected environment variable file is used with your test and the missing environment variable file is removed from the Missing Resources pane.

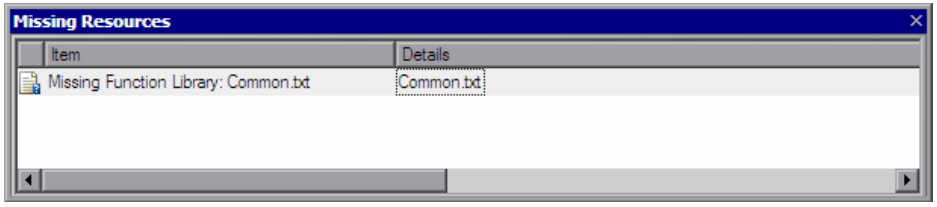
To remove a missing environment variable file:

Right-click the missing environment variable file you want to remove and select **Remove** from the context-sensitive menu. A confirmation message is displayed. Click **OK** to remove the missing environment variable. The missing environment variable file is removed from your test and from the Missing Resources pane.



Handling Missing Function Libraries

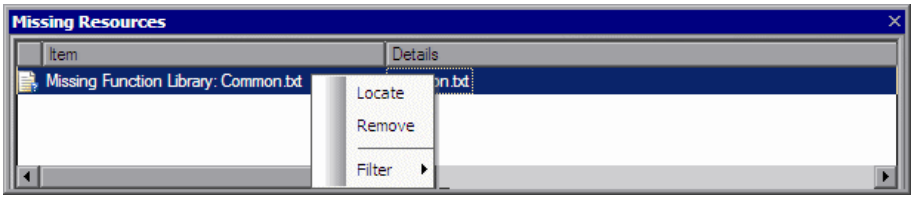
When you open a test that has associated function libraries, QuickTest verifies that the libraries you specified are accessible. If a function library cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your test.



The Missing Resources pane enables you to resolve a missing function library by locating or removing it.

To locate a missing function library:

1 Right-click the missing function library you want to locate and select **Locate** from the context-sensitive menu or double-click the missing function library you want to locate.

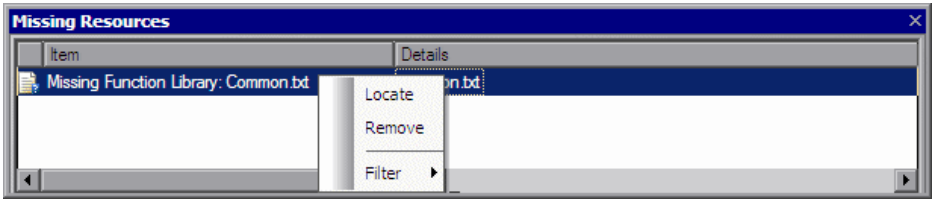


The Locate Function Library dialog box opens.

2 Browse to the function library you want to associate with your test and click **Open**. QuickTest associates the selected function library with your test and removes the missing function library from the Missing Resources pane.

To remove a missing function library:

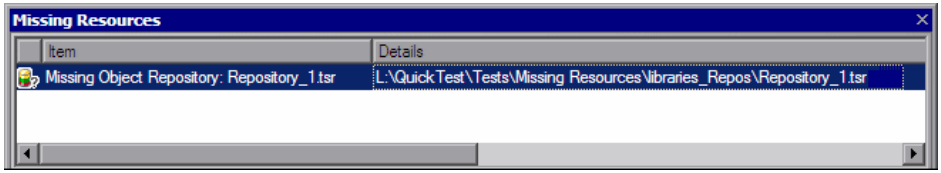
Right-click the missing function library you want to remove and select **Remove** from the context-sensitive menu. A confirmation message is displayed. Click **OK** to remove the function library. QuickTest removes the missing function library from your test and from the Missing Resources pane.



Note: Make sure that you handle any calls to functions in removed function libraries. When a function library is removed from your test, calls to those functions are not removed from your test.

Handling Missing Shared Object Repositories

When you associate a shared object repository with an action, QuickTest verifies that the repository you specified is accessible. In addition, QuickTest checks that all associated shared object repositories are accessible each time you open a test. If a shared object repository cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your test.



For example, if you modify the name of the shared object repository or the folder in which it is stored, you will need to map the shared object repository to the test.

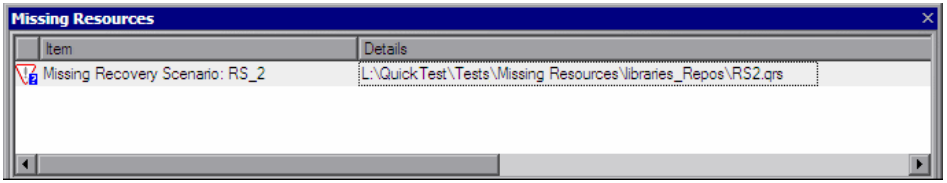
You can right-click the line displaying the missing object repository and select **Resolve** or double-click the line displaying the missing object repository and the Associate Repositories dialog box opens. The Associate Repositories dialog box enables you to associate one or more shared object repositories with one or more actions in your test. You can also remove object repository associations from selected actions, or from all actions in your test.

Note: You use the Associate Repositories dialog box to resolve a missing object repository by associating a new object repository with your test. The missing object repository will still be associated with your test and will still appear in the Missing Resources pane. To remove the missing object repository from the Missing Resources pane and your test, you must use the Remove Repository feature of the Associate Repository dialog box.

For more information, see “Managing Shared Object Repository Associations” on page 199.

Handling Missing Recovery Scenarios

When you open a test that has associated recovery scenarios, QuickTest verifies that the scenarios you specified are accessible. If a recovery scenario cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your test.



The Missing Resources pane enables you to resolve missing recovery scenarios by:

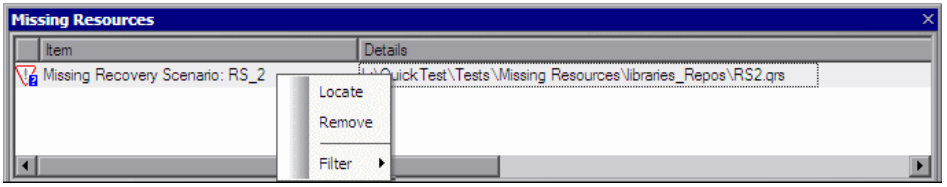
- Locating Missing Recovery Scenarios
- Removing Missing Recovery Scenarios

Locating Missing Recovery Scenarios

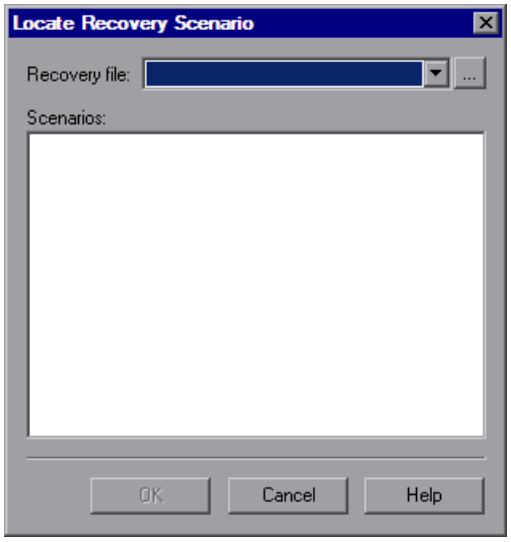
The Missing Resources pane enables you to locate missing recovery scenarios in your test. If your test contains more than one missing recovery scenario, when you locate the missing scenario in a recovery file, QuickTest may identify additional missing scenarios in that file. You can instruct QuickTest to locate these missing recovery scenarios simultaneously, or you can handle each missing scenario individually.

To locate a missing recovery scenario:

1 In the Missing Resources pane, right-click the recovery scenario you want to locate and select **Locate** from the context-sensitive menu or double-click the recovery scenario you want to locate.



The Locate Recovery Scenario dialog box opens.



2 Click the Browse button to select the recovery file. The **Scenarios** area displays all the scenarios contained in the selected recovery file.

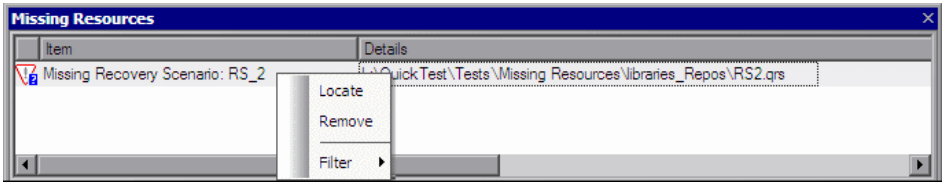
3 Select the missing recovery scenario from the list of recovery scenarios. Click **OK**. The selected recovery scenario is associated with your test and the missing recovery scenario is removed from the Missing Resources pane.

Note: If your test contains additional missing recovery scenarios that can be located in the same recovery file, QuickTest opens a message box asking you if you want to map these recovery scenarios as well. Click **Yes** to map all missing recovery scenarios, or click **No** to map only the scenario you specified.

Removing Missing Recovery Scenarios

You can remove a missing recovery scenario from a test if it is not needed.

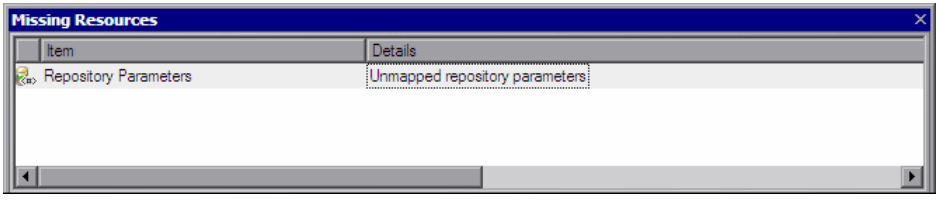
To remove a missing recovery scenario, in the Missing Resources pane, right-click the recovery scenario you want to remove and select **Remove** from the context-sensitive menu. A confirmation dialog is displayed. Click **OK** to remove the recovery scenario. The missing recovery scenario is removed from your test and from the Missing Resources pane.



Handling Unmapped Shared Object Repository Parameter Values

Every repository parameter used in your test must have a specified value. This can be a either a default value that was specified when the parameter was created, or it can be a value that you specify in your test. (For more information on repository parameters, see “Working with Repository Parameters” on page 228.)

When you open a test that uses an object repository with a repository parameter without a value, QuickTest indicates this by displaying **Repository Parameters** in the Missing Resources pane.



For example, suppose your application contains an edit box whose name property changes depending on a selection made in a previous screen. If you parameterized the value of the name property in the object repository using a repository parameter, but a default value was not defined for the repository parameter, you need to define a value for it. You can map it to a DataTable, an environment variable, a random number, or a test or action parameter. You can also define a constant value for it, and so forth.

If you right-click the line displaying **Repository Parameters** and select **Resolve** or double-click the line displaying **Repository Parameters**, the Map Object Repository Parameters dialog box opens, enabling you to specify values for any unmapped object repository parameter. You can filter the dialog box to display only unmapped parameters or all of the parameters in your test or the specific action (with mapped and unmapped values). For more information, see “Mapping Repository Parameter Values” on page 202.

42

Working with Data Tables

QuickTest enables you to insert and run steps that are driven by data stored in the Data Table.

This chapter includes:

About Working with Data Tables

The data your test uses is stored in the **design-time** Data Table, which is displayed in the Data Table pane while you insert and edit steps.

To view or hide the Data Table pane, select **View** > **Data Table** or click the **Data Table** toolbar button.

The Data Table has the characteristics of a Microsoft Excel spreadsheet, meaning that you can store and use data in its cells and you can also perform mathematical formulas within the cells.

You can use the Data Table provided with QuickTest, or you can use any Microsoft Excel (**.xls**) file. You configure the location of the Data Table in Resources pane of the Test Settings dialog box (**File** > **Settings** > **Resources** node).

You can use the DataTable, DTSheet and DTParameter utility objects to manipulate the data in any cell in the Data Table. For more information on these objects, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

Note: The use of complex and/or nested formulas in the Data Table is not supported.

You can insert Data Table parameters and output values into your test. Using Data Table parameters and/or output values in a test enables you to create a **data-driven** test or action that runs several times using the data you supply. In each repetition, or **iteration**, QuickTest uses a different value from the Data Table.

During the run session, QuickTest creates a **run-time** Data Table—a live version of the Data Table associated with your test. During the run session, QuickTest displays the run-time data in the Data Table pane so that you can see any changes to the Data Table as they occur.

When the run session ends, the run-time Data Table closes, and the Data Table pane again displays the stored design-time Data Table. Data entered in the run-time Data Table during the run session is not saved with the test. The final data from the run-time Data Table is displayed in the **Run-Time Data Table** in the Test Results window. For more information on the run-time Data Table, see "Viewing the Run-Time Data Table" on page 1056.

Tip: If it is important for you to save the resulting data from the run-time Data Table, you can insert a DataTable.Export statement to the end of your test to export the run-time Data Table to a file. You can then import the data to the design-time Data Table using the Data Table **File** > **Import From File** menu. Alternatively you can add a DataTable.Import statement to the beginning of your test to import the run-time Data Table that was exported at the end of the previous run session. For more information on these methods, see the *HP QuickTest Professional Object Model Reference*.

Working with Global and Action Sheets

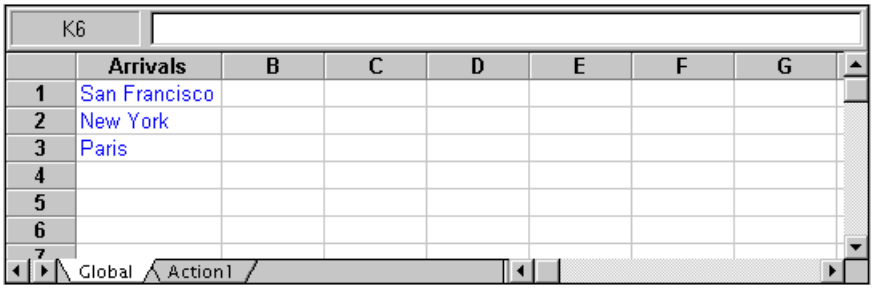
When working with tests, the Data Table has two types of data sheets—**Global** and **Action**. You can access the different sheets by clicking the appropriate tabs below the Data Table.

- You store data in the Global tab when you want it to be available to all actions in your test and you want the data to control the number of test iterations.
- You store data in the action's tab when you want to use the data in Data Table parameters for that action only and you want the data to control the number of action iterations.

  For example, suppose you are creating a test on the sample Mercury Tours Web site. You might create one action for logging in, another for booking flights, and a third for logging out. You may want to create a test in which the user logs onto the site once, and then books flights for five passengers. The data about the passengers is relevant only to the second action, so it should be stored in the action tab corresponding to that action.

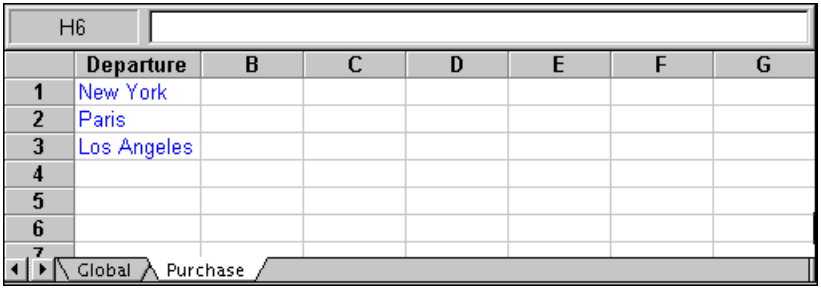
Global Sheet

The Global sheet contains the data that replaces parameters in each iteration of the test. If you create a Global parameter called Arrivals, the Global sheet might look like this:



Action Sheets

Each time you add a new action to the test, a new **action sheet** is added to the Data Table. Action sheets are automatically labeled with the exact name of the corresponding action. The data contained in an action sheet is relevant for Data Table parameters in the corresponding action only. For example, if a test had the Data Table below, QuickTest would use the data contained in the Purchase sheet when running iterations on action parameter steps within the **Purchase** action.



For more information on creating global and action parameters, see Chapter 24, “Parameterizing Values.”

Saving the Data Table

The Data Table contains the values that QuickTest substitutes for Data Table parameters when you run a test, as well as any other values or formulas you enter. Whenever you save your test, QuickTest automatically saves its Data Table as an **.xls** file.

When working with tests, the Data Table is saved with your test by default. You can save the Data Table in another location and instruct the test to use this Data Table when running a test. You specify a name and location for the Data Table in the Resources pane of the Test Settings dialog box.

For more information on the Test Settings dialog box, see Chapter 45, "Setting Options for Individual Tests."

Saving the Data Table in a specified location can be useful in the following circumstances:

- You want to run the same test with different sets of input values. For example, you can test the localization capabilities of your application by running your test with a different Data Table file for each language you want to test. You can also vary the user interface strings that you check in each language by using a different environment parameter file each time you run the test. For more information, see Chapter 24, "Parameterizing Values."
- You need the same input information for different tests. For example, you can test a Web version and a standard Windows version of the same application using different tests, but the same Data Table file.

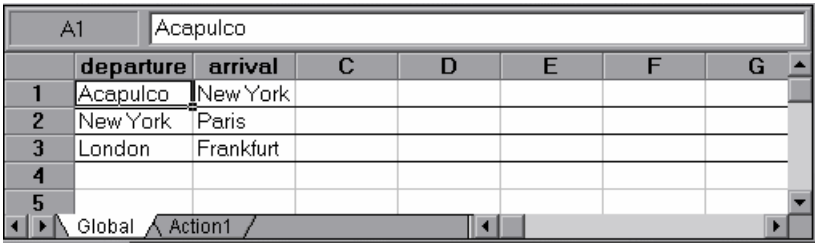
Note: If you select an external file as your Data Table, you must make sure that the column names in the external Data Table match the parameter names in the test and that the sheets in the external Data Table match the action names in the test.

Editing the Data Table

You can edit information in the Data Table by typing directly into the table cells. You use the Data Table in the same way as an Microsoft Excel spreadsheet, including inserting formulas into cells. You can also import data saved in Microsoft Excel, tabbed text file (.txt), or ASCII format.

For information on supported versions of Microsoft Excel, see the *HP QuickTest Professional Readme*.

To view or hide the Data Table pane, select **View** > **Data Table** or click the **Data Table** toolbar button.



Each **row** in the table represents the set of values that QuickTest submits for the parameterized arguments during a single iteration of the test or action. You define the number of iterations that an action runs in the Run tab of the Action Call Properties dialog box (**Edit** > **Action** > **Action Call Properties** > **Run** tab). You define the number of iterations that a test runs in the Run pane of the Settings dialog box (**File** > **Settings** > **Run** pane).

Each **column** in the table represents the list of values for a single parameterized argument. The column header is the parameter name.

You can also enter data and formulas in cells in the columns that are not intended for use with Data Table parameters (the columns that do not have a parameter name in the column header).

Guidelines for Working with the Data Table

- When you add data to the Data Table, you must enter the data in rows from top to bottom and left to right—you cannot leave a gap of an entire row or column. For example, if there is data in row 1, you cannot enter data in a cell in row 3 until you have entered data in row 2. Similarly, if there is data in column A, you cannot enter data in column C until you have entered data in column B.
- The value returned from the Data Table is always converted to a string. If you want the value to be converted to something other than a string, you can use VBScript conversion functions, such as CInt, CLng, CDbl, and so forth. For example:

```
Window("Flight Reservation").WinComboBox("Fly From:").Select
    CInt(DataTable("ItemNumber", dtGlobalSheet))
```

- When you add content to a Data Table cell, QuickTest changes the color of the row's bottom grid line from gray to black. When you run your test using the **Run on all rows** option (defined in **File** > **Settings** > **Run** pane, or **Edit** > **Action** > **Action Call Properties** > **Run** tab), QuickTest runs one iteration for each row whose bottom grid line is black.

  If you want to prevent QuickTest from running an iteration on a row when the **Run on all rows** option is selected, you must delete the entire row from the Data Table. To do this, select the row, right-click in the table, and select **Edit** > **Delete** from the data table's context menu (or use CTRL+K). (This restores the bottom grid line from black to gray.) Otherwise, if you use the **Clear** option from the table's **Edit** menu (or CTRL+X), or select a cell and press **Delete** on the keyboard, the data is deleted from the cells, but the row is not deleted and the black line remains. This means that QuickTest will run an iteration for this row even though there is no data in it.

Data Table Specifications

The main limitations for working with the Data Table are listed below:

- **Maximum worksheet size.** 65,536 rows by 256 columns
- **Column width.** 0 to 255 characters
- **Text length.** 16,383 characters
- **Formula length.** 1024 characters
- **Number precision.** 15 digits
- **Largest positive number.** 9.99999999999999E307
- **Smallest positive number.** 1E-307
- **Largest negative number.** -1E-307
- **Smallest negative number.** -9.99999999999999E307
- **Maximum number of names per workbook.** Limited by available memory
- **Maximum length of name.** 255
- **Maximum length of format string.** 255
- **Maximum number of tables (workbooks).** Limited by system resources (windows and memory)
- The use of colors and formatting in the Data Table is not supported.
- Complex and/or nested formulas are not supported in the Data Table.
- Combo box and list cells, conditional formatting, and other special cell formats are not supported in the Data Table.

Changing a Column Name

You can change the name of a column for a parameter by double-clicking the column heading cell. In the Change Parameter Name dialog box, you can type a new parameter name. This parameter name must be unique in the test. It can contain letters, numbers, commas, and underscores. The first character of the parameter name must be a letter or an underscore.

Note: If you change the name in the table, you must also change the name defined for the corresponding parameter in the test.

Using the Data Table Menu Commands

You can use the Data Table menu commands described below to edit the data in the Data Table. To open the Data Table menu, right-click a cell, a row heading or a column heading.

The following menus are available:

- File (described on page 1206)
- Sheet (described on page 1207)
- Edit (described on page 1207)
- Data (described on page 1209)
- Format (described on page 1209)

File Menu

The following commands are available in the **File** menu:

File Command	Description
Import From File	Imports an existing Microsoft Excel or tabbed text file into the Data Table. This command will import all the sheets in the selected Microsoft Excel file. If you want to import only one sheet from an existing Microsoft Excel file, use the **Sheet** > **Import** > **From File** command described below. **Notes:** ➤ The table file you import replaces all data in all sheets of the table. The first row in each Microsoft Excel sheet also replaces the column headers in the corresponding Data Table sheet. It is therefore essential that the first row of your Microsoft Excel sheet exactly matches the parameter names in your test, and that the file contains at least the same number of sheets as the current Data Table. ➤ If you import a Microsoft Excel table containing combo box or list cells, conditional formatting, or other special cell formats, the formats are not imported and the cells are displayed in the Data Table with a fixed value.
Export	Exports the table to a specified Microsoft Excel (**.xls**) file.
Print	Prints the entire table or the selected sheet.

Sheet Menu

The following commands are available in the **Sheet** menu:

Sheet Command	Description
Import > **From File**	Imports a tabbed text file or a single sheet from an existing Microsoft Excel file into the table. **Note:** The sheet you import replaces all data in the currently selected sheet of the table, and the first row in the Excel sheet replaces the column headers in the corresponding Data Table sheet. It is therefore essential that the first row of your Microsoft Excel sheet exactly matches the parameter names in your test.
Import > **From Database**	Imports data from the specified database to the current sheet.
Export	Exports the current sheet of the Data Table to a specified Microsoft Excel (**.xls**) file.

Edit Menu

The following commands are available in the **Edit** menu:

Edit Command	Description
Cut	Cuts the table selection and places it on the Clipboard.
Copy	Copies the table selection and places it on the Clipboard.
Paste	Pastes the contents of the Clipboard to the current table selection.
Paste Values	Pastes values from the Clipboard to the current table selection. Any formatting applied to the values is ignored. In addition, only formula results are pasted; formulas are ignored.
Clear	Clears formats or contents from the current selection. You can clear formats only, contents only (including formulas), or both formats and contents.

Edit Command	Description
Insert	Inserts empty cells at the location of the current selection. Cells adjacent to the insertion are shifted to make room for the new cells. Note that this option is available only when a row or column heading is selected.
Delete	Deletes the entire current row or column selection. Cells adjacent to the deleted cells are shifted to fill the space left by the vacated cells. Note that this option is available only when a row or column heading is selected.
Fill Right	Copies data in the left-most cell of a selected range to all the cells to the right of that left-most cell within the selected range.
Fill Down	Copies data in the top cell of a selected range to all cells below that top cell within the selected range.
Find	Finds a cell containing specified text. You can search by row or column in the table and specify to match case and/or find entire cells only. You can also search for formulas or values.
Replace	Finds a cell containing specified text and replaces it with different text. You can search by row or column in the table and specify to match case and/or to find entire cells only. You can also search for formulas or values. You can also replace all instances of the found text.
Go To	Goes to a specified cell. This cell becomes the active cell. You must enter the column and row number of the cell.

Data Menu

The following commands are available in the **Data** menu:

Data Command	Description
Recalc	Recalculates any formula cells in the table.
Sort	Sorts a selection of cells by row or column and keys in ascending or descending order.
AutoFill List	Opens the AutoFill Lists dialog box (described on page 1211).
Encrypt	Encodes the text in the selected cells. Note that you cannot decrypt data that has been encrypted. You can also use the Password Encoder to encrypt any text string. This can be useful for entering encrypted strings as method arguments in the Expert View. For more information, see "Inserting Encoded Passwords into Method Arguments and Data Table Cells" on page 406.

Format Menu

The following commands are available in the **Format** menu:

Format Command	Description
General	Sets format to General. The General format displays numbers with as many decimal places as necessary and no commas.
Currency(0)	Sets format to currency with commas and no decimal places. Note that QuickTest uses the currency symbol defined in your Windows Regional Settings dialog box.
Currency(2)	Sets format to currency with commas and two decimal places. Note that QuickTest uses the currency symbol defined in your Windows Regional Settings dialog box.
Fixed	Sets format to fixed precision with commas and no decimal places.
Percent	Sets format to percent with no decimal places. Numbers are displayed as percentages with a trailing percent sign (%).

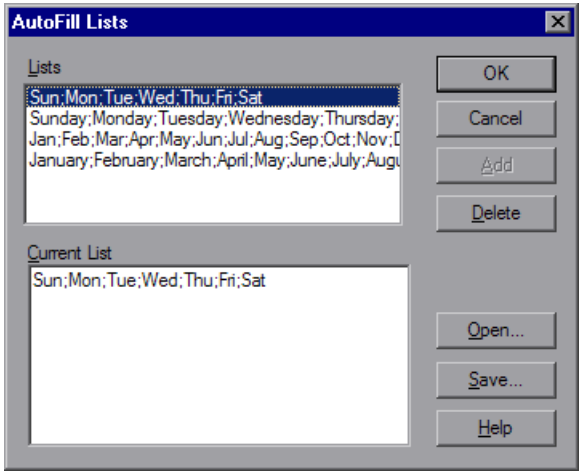
Format Command	Description
Fraction	Sets format to fraction in numerator denominator form, e.g. 1/2.
Scientific	Sets format to scientific notation with two decimal places.
date (dynamic)	Sets format to Date with the M/D/YY format.
Time: h:mm AM/PM	Sets format to Time with the h:mm AM/PM format.
Custom Number	Sets format to a custom number format that you specify. This option enables you to set special and customized formats for percentages, currencies, dates, times, and so forth.

You can also perform Data Table menu commands using shortcut keys. For more information, see "Performing QuickTest Commands" on page 46.

The AutoFill Lists Dialog Box

Description	Enables you to create, edit, or delete an autofill list. An autofill list contains frequently-used series of text such as months and days of the week. **To use an autofill list:** 1 Enter the first item into a cell in the table. 2 Drag the cursor, from the bottom right corner of the cell. QuickTest automatically fills in the cells in the range according to the autofill list.
How to Access	In the Data Table, right-click and select **AutoFill List**.
Learn More	**Primary task**: "About Working with Data Tables" on page 1197 **Additional related topics**: "AutoFill Lists Dialog Box Options" on page 1212

Below is an image of the AutoFill Lists dialog box:



AutoFill Lists Dialog Box Options

Option	Description
Lists	The lists that are available in your project. Four default lists are included.
Current List	The selected list. This pane can be used to create a new list. Separate the items in a new list with a semi-colon.
Add	Adds a new list to the **Lists** box.
Delete	Deletes a list from the **Lists** box.
Open	Opens the Open dialog box, in which you can browse to a previously created list.
Save	Opens the Save As dialog box, in which you can save a new list.

Using Data Table Files with Quality Center

When working with Quality Center and Data Tables, you must save the Data Table file in the Test Resources module in your Quality Center project before you specify the Data Table file in the Resources pane of the Test Settings dialog box.

You can add a new or existing Data Table file to your Quality Center project. Note that adding an existing Data Table file from the file system to a Quality Center project creates a copy of the file. Thus, once you save the file to the project, changes made to the Quality Center Data Table file will not affect the Data Table file in the file system and vice versa.

To use a Data Table file with Quality Center:

1. If you want to add a new Data Table file, create a new Microsoft Excel file in your file system with an **.xls** extension.
2. In Quality Center, create a new Data Table resource and then upload the **.xls** file you created in the previous step to the project's Test Resources module. For more information, see the *HP Quality Center User Guide*.

3 In the Test Settings dialog box, click the **Resources** node.

4 Select **Other location** and click the browse button to locate the Data Table file.

5 Create your test. When you save the test, QuickTest saves the Data Table file to the Quality Center project.

Importing Data from a Database

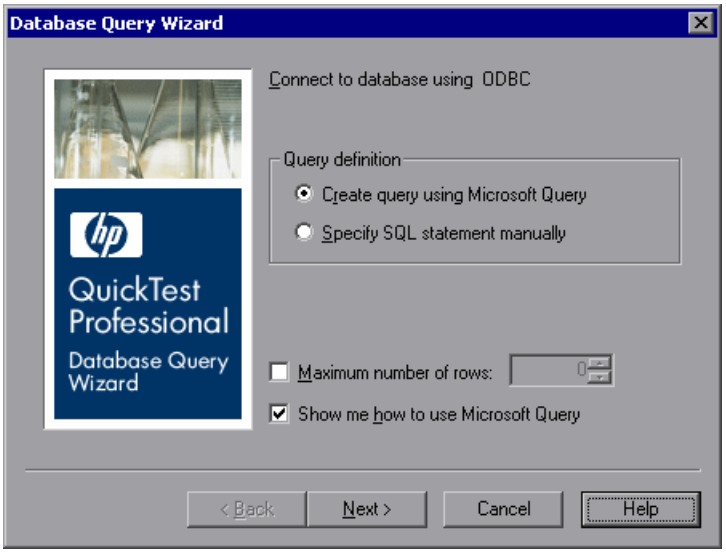
You can import data from a database by selecting a query from Microsoft Query or by manually specifying an SQL statement.

You can install Microsoft Query from the custom installation option of Microsoft Office.

Note: Contrary to importing an Excel file (**File > Import From File**), existing data in the Data Table is not replaced when you import data from a database. If the database you import contains a column with the same name as an existing column, the database column is added as a new column with the column name followed by a sequential number. For example, if your Data Table already contains a column called departures, a database column by the same name would be inserted into the Data Table as departures1.

To import data from a database:

1 Right-click on the Data Table sheet to which you want to import the data and select **Sheet** > **Import** > **From Database**. The Database Query Wizard opens.

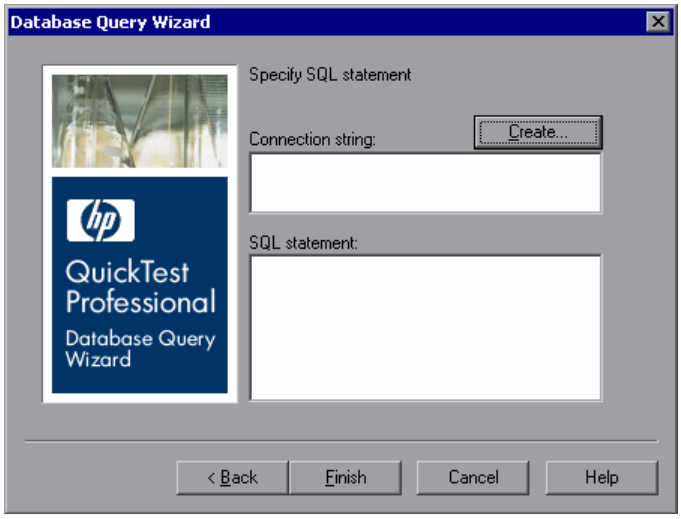


2 Select your database selection preferences and click **Next**. You can choose from the following options:

- **Create query using Microsoft Query.** Opens Microsoft Query, enabling you to create a new query. After you finish defining your query, you exit back to QuickTest. This option is only available if you have Microsoft Query installed on your computer.
- **Specify SQL statement manually.** Opens the **Specify SQL statement** screen in the wizard, which enables you to specify the connection string and an SQL statement. For more information, see step 3.
- **Maximum number of rows.** Select this check box and enter the maximum number of database rows to import. You can specify a maximum of 32,000 rows.
- **Show me how to use Microsoft Query.** Displays an instruction screen before opening Microsoft Query when you click **Next**. (Enabled only when **Create query using Microsoft Query** is selected).

3 If you chose **Create query using Microsoft Query** in the previous step, Microsoft Query opens. Choose a data source and define a query. For more information on creating a query, see "Creating a Query in Microsoft Query" on page 1216.

If you chose **Specify SQL statement manually** in the previous step, the following screen opens:



Specify the connection string and the SQL statement and click **Finish**.

- **Connection string.** Enter the connection string or click **Create** to open the ODBC Select Data Source dialog box. You can select a **.dsn** file in the ODBC Select Data Source dialog box or create a new **.dsn** file to have it insert the connection string in the box for you.
- **SQL statement.** Enter the SQL statement.

4 QuickTest takes several seconds to capture the database query and restore the QuickTest window. The resulting data from the database query is displayed in the Data Table.

Creating a Query in Microsoft Query

You can use Microsoft Query to choose a data source and define a query within the data source. For information on supported versions of Microsoft Query, see the *HP QuickTest Professional Readme*.

To choose a data source and define a query in Microsoft Query:

1 When Microsoft Query opens during the **Import data from database** process, choose a new or an existing data source.

2 Define a query.

3 In the Finish screen of the Query Wizard, select **Exit and return to QuickTest** and click **Finish** to exit Microsoft Query.

Alternatively, click **View data or edit query in Microsoft Query** and click **Finish**. After viewing or editing the data, select **File > Exit and return to QuickTest** to close Microsoft Query and return to QuickTest.

For more information on working with Microsoft Query, see the Microsoft Query documentation.

Using Formulas in the Data Table

You can use Microsoft Excel formulas in your Data Table. This enables you to create contextually relevant data during the run session. You can also use formulas as part of a checkpoint to check that objects created on-the-fly (dynamically generated) or other variable objects in your Web page or application have the values you expect for a given context.

When you use formulas in a Data Table to compare values (generally in a checkpoint), the values you compare must be of the same type, for example integers, strings, and so forth. When you extract values from different places in your applications using different functions, the values may not be of the same type. Although these values may look identical on the screen, a comparison of them will fail, since, for example, 8.2 is not equal to "8.2".

Note: The use of complex and/or nested formulas in the Data Table is not supported.

You can use the TEXT and VALUE functions to convert values from one type to another as follows:

- TEXT(value, format) returns the textual equivalent of a numeric value in the specified format, so that, for example the formula =TEXT(8.2, "0.00") is "8.20".
- VALUE(string) returns the numeric value of a string, so that, for example, =VALUE("$8.20") is 8.20.

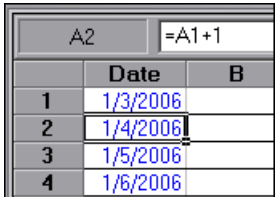
For more information on using worksheet functions, see the Microsoft Excel documentation.

Using Formulas to Create Parameterization Data

You can enter formulas rather than fixed values in the cells of a parameter column.

For example, suppose you want to parameterize the value for a WebEdit object that requires a date value no earlier than today's date. You can set the cells in the **Date** column to the date format, and enter the =NOW() Excel formula into the first row to set the value to today's date for the first iteration.

Then you can use another formula in the remaining rows to enter the above date plus one day, as shown below. By using this formula you can run the test on any day and the dates will always be valid.

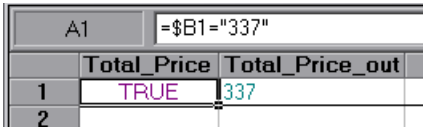


For more information on using parameters, see Chapter 24, "Parameterizing Values."

Using Formulas in Checkpoints

You can use a formula in a checkpoint to confirm that an object created on-the-fly (dynamically generated) or another variable object in your Web page or application contains the value it should for a given context. For example, suppose a shopping cart Web site displays a price total. You can create a text checkpoint on the displayed total value and use a Data Table formula to check whether the site properly computes the total, based on the individual prices of the products selected for purchase in each iteration.

When you use the Data Table formula option with a checkpoint, QuickTest creates two columns in the Data Table. The first column contains a default checkpoint formula. The second column contains the value to be checked in the form of an output parameter. The result of the formula is Boolean—TRUE or FALSE.



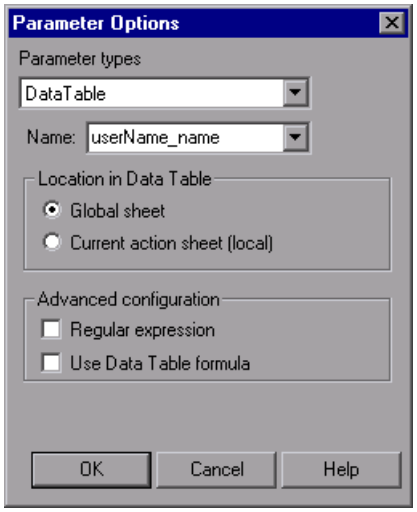
A FALSE result in the checkpoint column during a test run causes the test to fail.

After you finish adding the checkpoint, you can modify the default formula in the first column to perform the check you need.

To use a formula in a checkpoint:

1. Select the object or text for which you want to create a checkpoint and open the Insert Checkpoint dialog box as described in Chapter 17, "Understanding Checkpoints."
2. In the **Configure value** area, click **Parameter**.

3 Click the **Parameter Options** button. The Parameter Options dialog box opens.



4 Select **Data Table** as the parameter type and choose a parameter from the **Parameter name** box list or enter a new name.

- To use an existing parameter, select it from the list.
- To create a new parameter, either use the default parameter name or enter a descriptive name for the parameter.

5 Select the **Use Data Table formula** check box and click **OK** to close the Parameter Options dialog box.

Note: You cannot select **Use Data Table formula** if **Regular expression** is selected.

---

6 Specify your other checkpoint setting preferences as described in Chapter 17, "Understanding Checkpoints."

7 Click **OK.** The two columns are added to the table, and the checkpoint step is inserted into your test.

8 Highlight the value in the first (formula) column to view the formula and modify the formula to fit your needs.

9 If you want to run several iterations, add the appropriate formula in subsequent rows of the formula column for each iteration in the test or action.

Tip: You can encode passwords to use the resulting strings as method arguments or Data Table parameter values. For more information, see "Inserting Encoded Passwords into Method Arguments and Data Table Cells" on page 406.

You can also encrypt strings in Data Table cells using the Encrypt option in the Data Table menu. For more information, see "Data Menu" on page 1209.

Using Data Table Scripting Methods

QuickTest provides several Data Table methods that enable you to retrieve information on the run-time Data Table and to set the value of cells in the run-time Data Table. You enter these statements manually in the Expert View. For more information on working in the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows."

From a programming perspective, the Data Table is made up of three types of objects—DataTable, DTSheet (sheet), and DTParameter (column). Each object has several methods and properties that you can use to retrieve or set values.

For more details on the Data Table methods, see the *HP QuickTest Professional Object Model Reference*.

43

Working with Process Guidance

Process guidance is a tool that provides procedures and descriptions on how to best perform specific processes. You use process guidance to learn about new processes and to learn the preferred methodology for performing processes with which you are already familiar. For this reason, process guidance is applicable to both new and experienced users.

A process is a collection of activities, or sub-processes. Each process walks you step-by-step through the activities that are required for that process. As you navigate through the activities for each process and perform the tasks described in each activity, you become acquainted with the way in which a particular process should be performed.

QuickTest provides a built-in package that comprises several processes. These processes provide introductory information and tips on how to perform the most common QuickTest tasks, such as planning and creating a test.

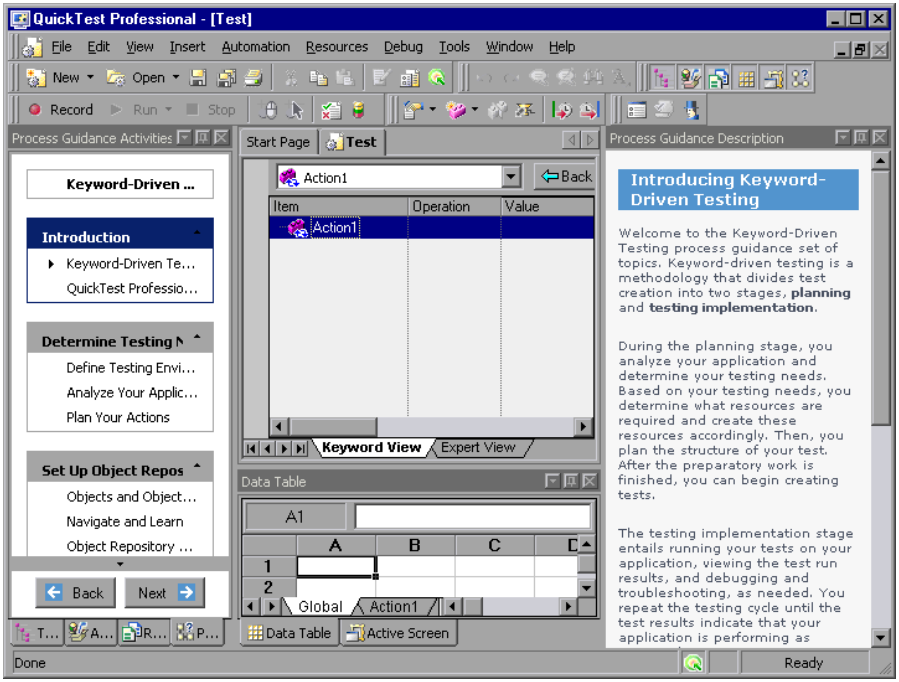
Your organization can also create its own custom processes to guide users through specific requirements and best practices relevant to your organization. For more information, see "Creating Custom Process Guidance Packages" on page 1569.

This chapter includes:

Process Guidance Panes

In QuickTest, process guidance is displayed in two panes: the **Process Guidance Activities** pane and the **Process Guidance Description** pane.

You display or hide these panes by choosing **View** > **Process Guidance** or clicking the **Process Guidance panes** toggle button.



Process Guidance Activities Pane

The **Process Guidance Activities** pane (shown on the left) lists the activities that are part of the selected process. Activities are often grouped, enabling you to navigate directly to the sub-process that interests you. The example above illustrates some of the groups and activities in the Keyword-Driven Testing process. For example, the Determine Testing Needs group contains three activities: Define Testing Environment, Analyze Your Application, and Plan Your Actions.

In the **Process Guidance Activities** pane, you can:

- Click an activity to open the relevant topic in the **Process Guidance Description** pane.
- Check which activity is displayed in the **Process Guidance Description** pane. (An arrow points to the currently selected activity.)
- Use the **Back** and **Next** buttons to navigate up and down between activities and to display the topic for the previous or next activity in the **Process Guidance Description** pane.
- Position the cursor over the **Up** and **Down** arrows to scroll through the list of activities. (The up arrow is located directly below the Process Guidance Activities title bar; the down arrow is located directly above the **Back** and **Next** buttons.)

Process Guidance Description Pane

The **Process Guidance Description** pane (shown on the right in the example above) displays the topic description, for the selected activity.

Each description introduces you to a specific activity and provides links to locations in which you can find more information about how to perform that activity. Additionally, many of the descriptions include interactive links that open dialog boxes or other relevant features, enabling you to directly access the features that are being described.

Opening Process Guidance

You can open a process from the Start Page, from the **Automation** menu, or from the Process Guidance Activities pane.

Start Page

The **Process Guidance List** on the Start Page displays all available processes. Some processes may be available only under certain conditions. For example, the Business Components process guidance is available only if you are connected to a Quality Center project that supports business process testing. Additionally, some processes may be visible only if you have a specific add-in loaded. For example, the **Testing SAP GUI for Windows** built-in process is visible only if the SAP add-in is loaded.

When you select a QuickTest process from the list, the relevant document type opens. For example, if a test document is open and you select the **Application Areas** process, a new application area opens, enabling you to navigate through the application area as you navigate through the selected process (provided that you are connected to a Quality Center project with business process testing support).

To open a specific process from the Start Page:

1. In QuickTest, click the **Start Page** tab to display the Start Page. (If the Start Page tab is not visible, select **View** > **Start Page** to open the Start Page.)
2. In the **Process Guidance List**, click the link for the process you want to open. The list of activities is displayed in the **Process Guidance Activities** pane, and a description of the first activity in the list is displayed in the **Process Guidance Description** pane.

Tip: If the **Process Guidance List** is empty, select **File** > **Process Guidance Management** and select at least one process in the Process Guidance Management dialog box. For more information, see "The Process Guidance Management Dialog Box" on page 1226.

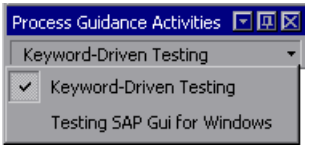
Automation Menu Command

You can open any process that is available for a currently open document type or for a loaded QuickTest add-in by choosing **Automation** > **Process Guidance List** and then choosing a process from the list.

If the **Process Guidance List** is empty, select **File** > **Process Guidance Management** and select at least one process in the Process Guidance Management dialog box. Then reopen the current document or open a new document to refresh the list of available processes in the **Process Guidance List** menu.

If you want to open a process that is not relevant for the current testing document or loaded QuickTest add-in, you need to open the process from **Process Guidance List** in the Start Page.

If the currently open testing document has more than one process available, you can navigate between these process by selecting the required process from the drop-down list in the process title.



Managing the List of Available Processes

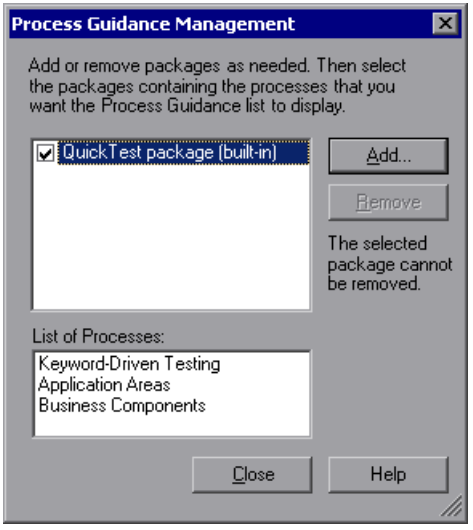
Processes are stored in process guidance packages. QuickTest provides a built-in package containing several processes. This package is listed by default in the Process Guidance Management dialog box.

Your organization may provide additional packages that include processes that are specific to your organization, your team, your role in your organization, and so on. For more information, see “Creating Custom Process Guidance Packages” on page 1569.

The Process Guidance Management Dialog Box

Description	Enables you to manage the list of processes that are available in QuickTest.
How to Access	**File** menu > **Process Guidance Management**
Important Information	➤ You cannot remove the built-in QuickTest package. ➤ You cannot include or exclude individual processes from within a package.
Learn More	**Conceptual overview**: "Working with Process Guidance" on page 1221 **Primary tasks**: ➤ "Including and Excluding Packages" on page 1227 ➤ "Adding Process Guidance Packages" on page 1228 **Additional related topics**: see "Additional References" on page 1227

Below is an image of the Process Guidance Management dialog box:



Process Guidance Management Dialog Box Options

Option	Description
Add	Enables you to add processes specific to your organization to the Process Guidance List on the Start Page.
Remove	Enables you to remove processes from the Process Guidance List on the Start Page.

Additional References

Related User Interface Topics	"Process Guidance Panes" on page 1222
Related Tasks	"Opening Process Guidance" on page 1224

Including and Excluding Packages

You can select to include or exclude a package in the set of packages available in QuickTest.

When you select to include a package, QuickTest adds all of the processes in that package to the **Process Guidance List** on the Start Page (excluding processes for QuickTest add-ins that are not currently loaded). The processes that are available for the currently open document type and for the currently loaded QuickTest add-ins are also added to the **Process Guidance List** in the **Automation** menu, and can be opened after you refresh the list by closing and reopening the current document or by opening a new document of the same type.

You cannot include or exclude individual processes from within a package.

To include or exclude a package in the set of packages available in QuickTest:

1. Select **File** > **Process Guidance Management**. The Process Guidance Management dialog box opens.
2. Select the check box adjacent to the package whose processes you want to include, or clear the check box adjacent to the package whose processes you want to exclude.
3. Click **Close**. QuickTest adds or removes the relevant processes in the **Process Guidance List**.

Adding Process Guidance Packages

If your organization has its own processes, you can add them to the **Process Guidance List** on the Start Page. You do this by adding the relevant package to the Process Guidance Management dialog box and selecting to show it.

To add a package to the list:

1. Select **File** > **Process Guidance Management**. The Process Guidance Management dialog box opens.
2. In the Process Guidance Management dialog box, click **Add**. The Open dialog box opens.
3. Browse to the process guidance package file and click **Open**. The package is added to the list of available packages.

Part IX

Configuring QuickTest Settings

44

Setting Global Testing Options

You can control how QuickTest works with tests by setting global testing options.

This chapter includes:

About Setting Global Testing Options

Global testing options affect both how you work with tests and the general appearance of QuickTest. For example, you can choose not to display the Start Page when QuickTest starts, or you can set the timing-related settings used by QuickTest when running a test. The values you set remain in effect for all tests and for subsequent testing sessions. You can set global testing options using the Options dialog box (described on page 1232) or by inserting statements in the Expert View.

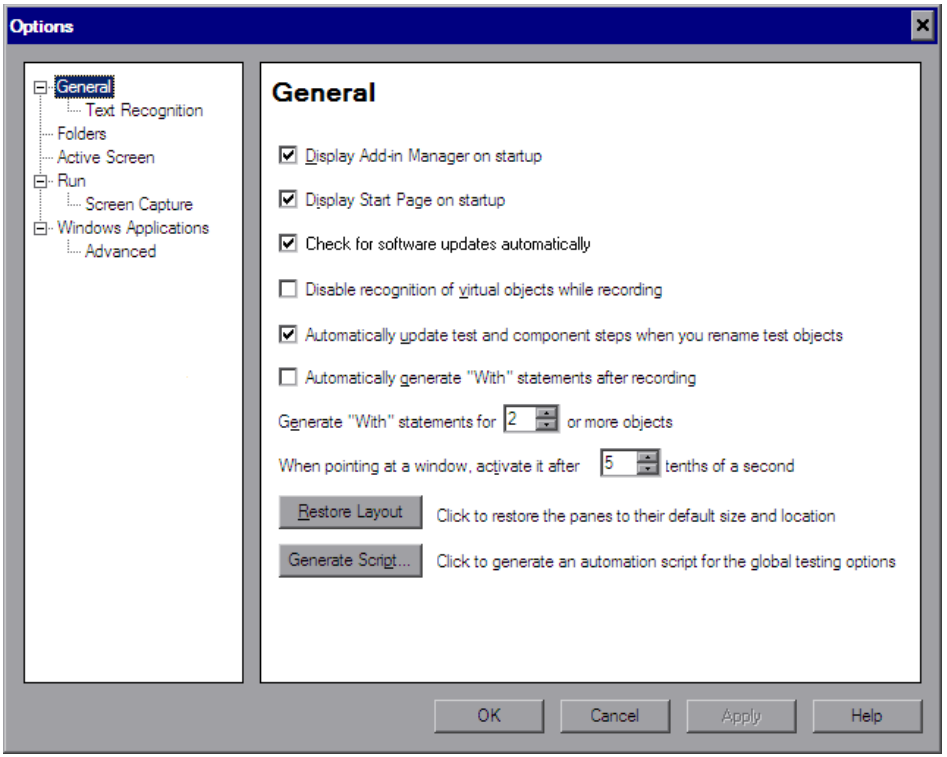
You can also set testing options that affect only the test currently open in QuickTest. For more information, see Chapter 45, "Setting Options for Individual Tests."

Using the Options Dialog Box

You can use the Options dialog box to modify your global testing options. The values you set remain in effect for all subsequent QuickTest sessions.

To set global testing options:

1 Select **Tools** > **Options** or click the **Options** toolbar button. The Options dialog box opens. It is divided into two parts: a navigation pane on the left and an options display pane on the right.



2 Select the required node from the navigation tree and set the options in the options display pane as necessary. For information on the available options in each node, see the table below.

3 Click **Apply** to apply your changes and keep the dialog box open, or click **OK** to save your changes and close the dialog box.

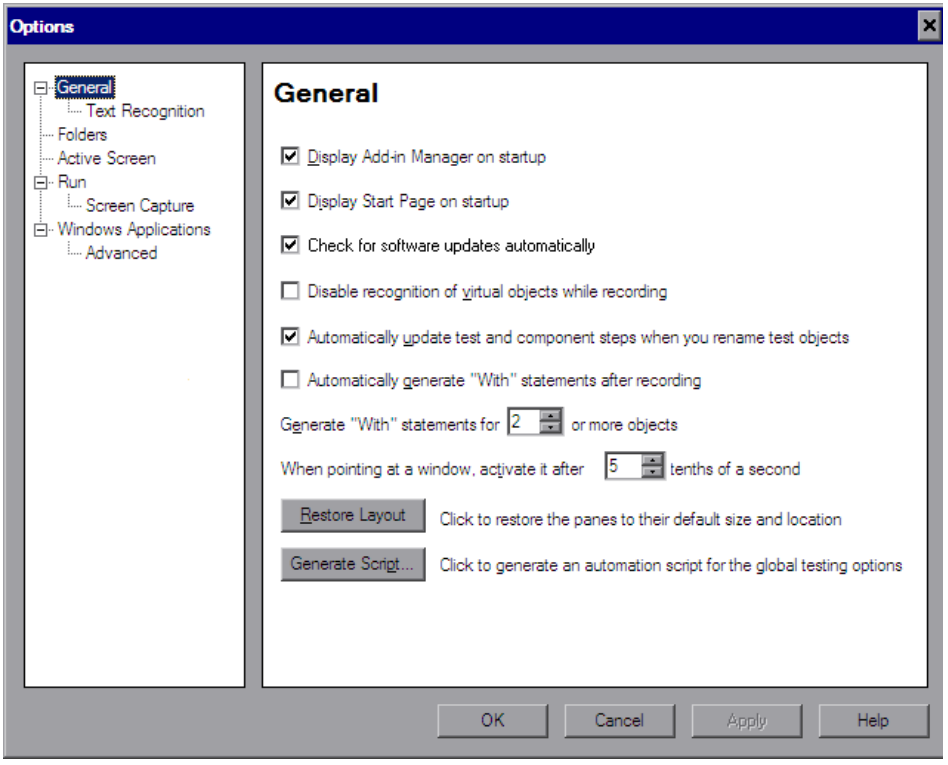
The navigation tree contains the following nodes:

Node	Options
General	Options for general test settings. For more information, see "Setting General Testing Options" on page 1234. The General node also contains the Text Recognition sub-node. For more information, see "Setting Text Recognition Options" on page 1236.
Folders	Options for entering the folders (search paths) in which QuickTest searches for tests, actions, or files that are specified as relative paths in dialog boxes and statements. For more information, see "Setting Folder Testing Options" on page 1237. **Note:** If you are working with the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.
Active Screen	Options for configuring which information QuickTest saves and displays in the Active Screen while recording. For more information, see "Setting Active Screen Options" on page 1240.
Run	Options for running tests. For more information, see "Setting Run Testing Options" on page 1253. The Run node also contains the Screen Capture node. For more information, see "The Options Dialog Box: Run > Screen Capture Pane" on page 1255.
Windows Applications	Options for configuring how QuickTest tests interface with Windows applications. The Windows Applications node also includes the Advanced node. For more information, see the section on testing Windows-based applications in the *HP QuickTest Professional Add-ins Guide*.

The navigation tree may contain additional nodes, depending on the add-ins that are currently loaded. For more information, see the relevant section in the *HP QuickTest Professional Add-ins Guide*.

Setting General Testing Options

The General pane options affect the general appearance of QuickTest and other general testing options.



The General node also contains the Text Recognition sub-node. For more information, see “The Options Dialog Box: General > Text Recognition Pane” on page 742.

The General pane includes the following options:

Option	Description
Display Add-in Manager on startup	Determines whether the Add-in Manager is displayed when starting QuickTest. For information on working with the Add-in Manager, see the section on loading QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.
Display Start Page on startup	Determines whether the Start Page is displayed when starting QuickTest.
Check for software updates Automatically	Instructs QuickTest to automatically check for software updates. For more information, see "Updating QuickTest Software" on page 18.
Disable recognition of virtual objects while recording	Determines whether the defined virtual objects stored in the Virtual Object Manager are recognized while recording. For more information, see Chapter 47, "Learning Virtual Objects."
Automatically update test and component steps when you rename test objects	Determines whether to automatically update test and component steps when you rename test objects in the local or shared object repository. For more information, see "Renaming Test Objects" on page 169.
Automatically generate "With" statements after recording	Instructs QuickTest to automatically generate With statements when you record. For more information, see "Generating With Statements for Your Test" on page 806.
Generate "With" statements for __ or more objects	Indicates the minimum number of identical, consecutive objects for which you want to apply the With statement. This setting is used when QuickTest automatically generates With statements after recording and when you select to generate With statements for an existing action. **Default = 2** For more information, see "Generating With Statements for Your Test" on page 806.

Option	Description
When pointing at a window, activate it after __ tenths of a second	Specifies the time (in tenths of a second) that QuickTest waits before it sets the focus on an application window when using the pointing hand to point to an object in the application (for Object Spy, checkpoints, Step Generator, Recovery Scenario Wizard, and so forth). **Default = 5**
Restore Layout	Restores the layout of the QuickTest window so that it displays the panes and toolbars in their default sizes and positions. **Note:** QuickTest recalls your most recent window layout for each of its operating modes: view/edit, record, and run. For more information, see "Customizing the QuickTest Window Layout" on page 1144.
Generate Script	Generates an automation script containing the current global testing options. For more information, see "Automating QuickTest Operations" on page 1403 or the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

Setting Text Recognition Options

The Text Recognition node of the navigation tree displays the General > Text Recognition pane, which enables you to configure how QuickTest identifies text in your application. For more information, see "The Options Dialog Box: General > Text Recognition Pane" on page 742.

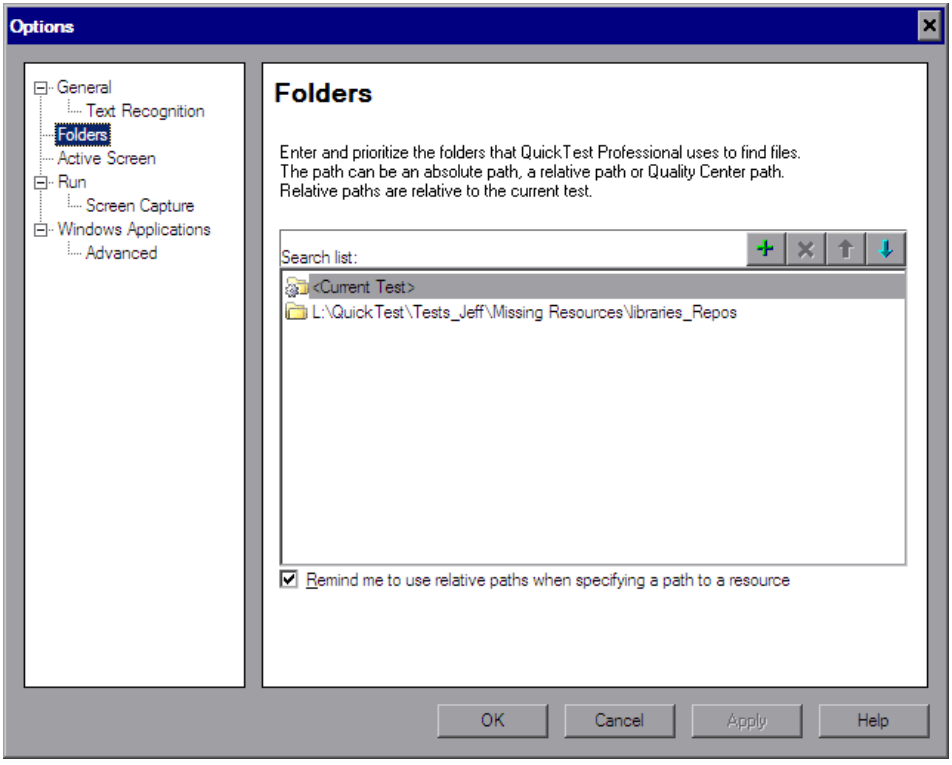
Setting Folder Testing Options

The Folders pane enables you to enter the folders (search paths) in which QuickTest searches for tests, actions, or resource files that are specified as relative paths in dialog boxes and steps. For example, suppose you add the folder in which all of your tests are stored to the folders list. If you later insert a copy of an action to a test, you only have to enter the name of the test containing the action you want to insert in the Insert Copy of Action dialog box. QuickTest searches for the test's path in the folders you specified in the Folders pane.

Notes:

- The current test is listed in the **Search list** by default. It cannot be deleted.
- If you are working with the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.
- For more information on relative or absolute paths, see "Using Relative Paths in QuickTest" on page 316.

QuickTest searches for the specified test, action, or file in the order in which the folders are displayed in the search list. If the same file name exists in more than one folder, QuickTest uses the first instance it finds.



The Folders pane includes the following options:

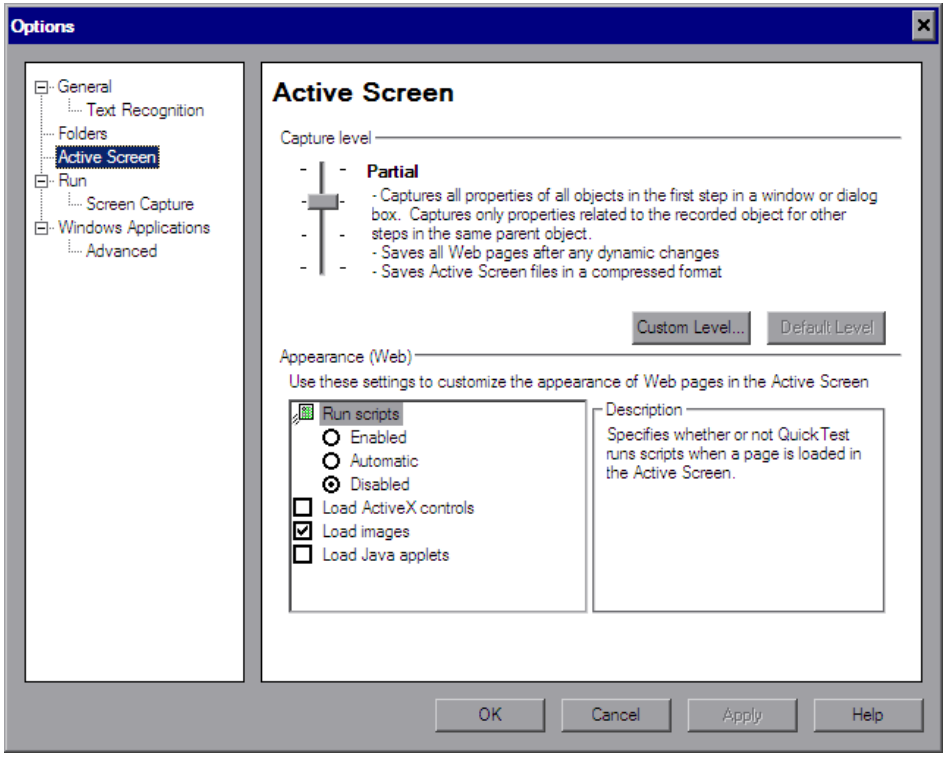
Option	Description
Search list	Indicates the folders in which QuickTest searches for tests, actions, or files. If you define folders here, you do not need to designate the full path of a test, action, or file in other dialog boxes or call statements. The order of the search paths in the list determines the order in which QuickTest searches for a specified action or file.
	Adds a new folder to the search list. **Tips:** ➤ To add a Quality Center path when connected to Quality Center, click this button. QuickTest adds [QualityCenter], and displays a browse button so that you can locate the Quality Center path. ➤ When not connected to Quality Center, hold the SHIFT key and click this button. QuickTest adds [QualityCenter], and you enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [QualityCenter]. For example: [QualityCenter] Subject\Tests. ➤ Note that QuickTest searches Quality Center project folders only when you are connected to the corresponding Quality Center project.
	Deletes the selected folder from the search list.
	Moves the selected folder up in the list.
	Moves the selected folder down in the list.
Remind me to use relative paths when specifying a path to a resource	When saving a resource, you can choose to be prompted to use a relative path. For more information, see "Using Relative Paths in QuickTest" on page 316. **Note:** When QuickTest is connected to a Quality Center 10.00 project, a reminder is displayed only if you select a path in the file system or in a Quality Center 9.x project.

Tip: You can use a PathFinder.Locate statement in your test to retrieve the complete path that QuickTest will use for a specified relative path based on the folders specified in the Folders pane. For more information, see the *HP QuickTest Professional Object Model Reference*.

Setting Active Screen Options

Description	Enables you to specify which information QuickTest saves and displays in the Active Screen while recording and running tests. The more information saved in the Active Screen, the easier it is to edit the test after it is recorded. However, more information saved in the Active Screen adds to the recording time and disk space required. This is especially critical with Windows-based add-ins, as they require more disk space to save Active Screen data.
How to Access	**Tools** menu > **Options** item > **Active Screen** node.
Important Information	When you are recording on an MDI (Multiple Document Interface) application, the Active Screen does not capture information for non-active child frames.
Learn More	**Conceptual overview**: "Working with the Active Screen" on page 376 **Primary task:** "Increasing or Decreasing the Active Screen Information Saved with a Test" on page 378 **Additional related topics**: "Additional References" on page 1244

Below is an image of the Active Screen pane in the Options dialog box:



Options Dialog Box: Active Screen Pane Options

Option	Description
Capture level	Specifies the objects for which QuickTest stores data in the Active Screen. Use the slider to select one of the following options: ➤ **Complete.** Captures all properties of all objects in the application's active window/dialog box/Web page in the Active Screen of each step. This level saves Web pages after any dynamic changes and saves Active Screen files in a compressed format. ➤ **Partial.** (Default.) Captures all properties of all objects in the application's active window/dialog box/Web page in the Active Screen of the first step performed in an application's window, plus all properties of the recorded object in subsequent steps in the same window. This level saves Web pages after any dynamic changes and saves Active Screen files in a compressed format. ➤ **Minimum.** Captures properties only for the recorded object and its parent in the Active Screen of each step. This level saves the original source HTML of all Web pages (prior to dynamic changes) and saves Active Screen files in a compressed format. ➤ **None.** Disables capturing of Active Screen files for all applications and Web pages.
Custom Level	Enables you to specify custom Active Screen options. For more information, see "The Custom Active Screen Capture Settings Dialog Box" on page 1244.
Default Level	Returns the capture level settings to the predefined default level (**Partial**).

Option	Description
Appearance (Web)	Enables you to modify how QuickTest displays captured Web pages in the Active Screen. ➤ **Run scripts.** Specifies whether QuickTest runs scripts when a page is loaded in the Active Screen, according to one of the following options: ➤ **Enabled.** Runs scripts whenever loading a page in the Active Screen. ➤ **Automatic.** Runs scripts as needed, according to the page that is displayed. ➤ **Disabled.** Prevents scripts from running when loading a page in the Active Screen. **Note:** This option refers only to scripts that run automatically when the page loaded. It does not enable you to activate scripts in the Active Screen by performing an operation on the screen. ➤ **Load ActiveX controls.** Instructs QuickTest to load ActiveX controls from your browser page to the Active Screen, so that for each step you can preview how the page is actually displayed in the application. If this option is cleared, a default ActiveX image is displayed in the Active Screen for all ActiveX control objects. ➤ **Load images.** Instructs QuickTest to load images from your browser page to the Active Screen. ➤ **Load Java applets.** Instructs QuickTest to load Java applets from your browser page to the Active Screen, so that for each step you can preview how the page is actually displayed in the application. If this option is cleared, a default Java image is displayed in the Active Screen for all Java applet objects. **Notes:** ➤ QuickTest loads ActiveX controls or Java applets to the Active Screen in view-only mode. You cannot perform operations or retrieve additional information on the loaded ActiveX or Java objects. To perform operations on these items from the Active Screen, you must load the relevant add-in and then record directly on the ActiveX or Java object. ➤ ActiveX controls or Java applets that are loaded to the Active Screen may not work exactly as they do in the application. In some cases, this may cause unexpected behavior, depending on the implementation of the specific controls or applets that are loaded.

Additional References

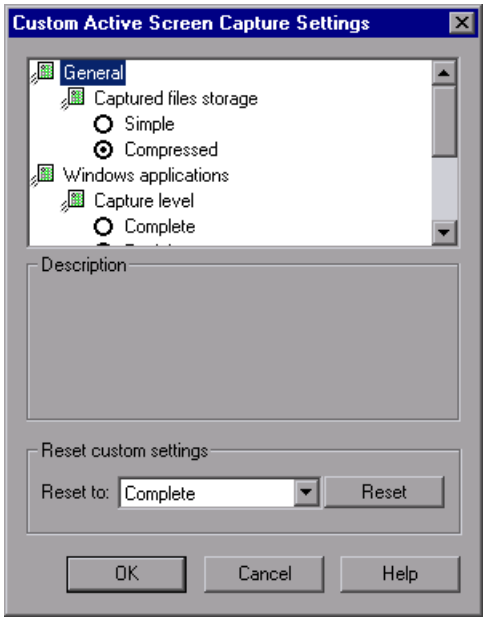
Related User Interface Topics	"The Custom Active Screen Capture Settings Dialog Box" on page 1244
Related Tasks	➤ "Updating a Single Active Screen Capture" on page 380 ➤ "Updating all Active Screen Captures in a Test Using Update Run Mode" on page 1125
Other Related Information	"Tips for Improving Active Screen Performance" on page 382

The Custom Active Screen Capture Settings Dialog Box

Description	Enables you to customize how QuickTest captures and saves Active Screen information. When you apply custom Active Screen settings, you override the capture-level setting in the Active Screen pane with all of the settings in the Custom Active Screen Capture Settings dialog box.
Accessed by	**Tools** menu> **Options** dialog box > **Active Screen** node > **Custom Level** button

Important Information	The default settings in the Custom Active Screen Capture Settings dialog box do not reflect the selected capture-level setting in the Active Screen pane of the Options dialog box. If you want to customize only specific settings, use the **Reset to** option to ensure that all other settings are using the capture-level setting you prefer and then modify the specific settings you need.
Related Tasks	➤ You can specify whether to save screen captures of the Active Screen using the **Save Active Screen files** option in the Save dialog box. See "Saving a Test" on page 324. ➤ You can use the **Update Run Mode** option to modify the amount of information saved in the Active Screen after you modify the Active Screen capture settings. See "Updating a Test Using the Update Run Mode Option" on page 1125.

Below is an image of the Custom Active Screen Capture Settings dialog box:



Note: The Custom Active Screen Capture Settings dialog box may also contain options applicable to any QuickTest add-ins installed on your computer.

Custom Active Screen Capture Settings Dialog Box Options

Option	Description
Settings box	Enables you to select the specific setting options that determine how QuickTest captures and saves Active Screen information. The Settings box may also contain options applicable to QuickTest add-ins installed on your computer. See: ➤ "General Options" on page 1247 ➤ "Capture Level Options" on page 1247 ➤ "Web Options" on page 1252
Description	Provides a description of the option selected in the Settings box.
Reset custom settings	Enables you to reset the custom settings to one of the predefined levels provided with QuickTest (**Complete**, **Partial**, **Minimum**, or **None**) by choosing a level from the **Reset to** list and clicking the **Reset** button. For more information on the available capture levels, see "Capture Level Options" on page 1247.

General Options

By selecting a **Captured files storage** option, you can specify the type of compression QuickTest uses for storing captured Active Screen information.

- **Simple.** Instructs QuickTest to save Active Screen captures in standard uncompressed file formats (for example, **.html** and **.png**).
- **Compressed.** Instructs QuickTest to save Active Screen captures in a compressed (zipped) file format. Using this option saves disk space, but it may affect the time it takes to load images in the Active Screen. This is the default option.

Capture Level Options

The Custom Active Screen Capture Settings dialog box enables you to customize how QuickTest captures and saves Active Screen information.

Capture level options are available for Java applets or applications, SAP GUI for Windows applications, Oracle applications, Windows-based applications, and Terminal Emulator applications. The options available in the Custom Active Screen Capture Settings dialog box depend on the add-ins that are installed.

By selecting a **Capture level** option, you can specify which properties are captured for each object in an application when it is captured for the Active Screen.

Depending on your testing requirements, you can choose between different levels of Active Screen capture. However, you should take into consideration that the less information captured for the Active Screen, the better the performance.

For example, if you select the **Complete** capture level option, you can add checkpoints on every test object displayed in any Active Screen capture after recording, but it will take more time and use more disk space to record a single operation. Selecting **Partial** enables QuickTest to record faster and use less disk space, but there may be limitations on the operations you can perform from the Active Screen after recording.

The following sections describe the capture level options available for different environments.

Java Applications or Applets

The following **Capture level** options are available for Java applications or Java applets:

- **Complete.** Instructs QuickTest to save all identification properties of all objects in the application or applet's open window/dialog box in the Active Screen of each step.
- **Partial.** (Default) Instructs QuickTest to save all identification properties of all objects in the application or applet's open window/dialog box in the Active Screen of the first step performed in that window, plus all properties of the recorded object only, in subsequent steps in the same window.
- **Minimum.** Instructs QuickTest to save all identification properties for the recorded object plus all identification properties for the parent objects in the recording hierarchy.
- **None.** Disables capture of Active Screen files for Java applications or Java applets.

When the **Complete** or **Minimum** capture level is selected, the following setting in the Custom Active Screen Capture Settings dialog box is also relevant for Java applications or Java applets:

Disable capture of the following objects. Prevents QuickTest from capturing the data of steps performed on other objects for the selected test object types in the Active Screen. These objects will be visible in the Active Screen as images only.

By default, **JavaObject** and **JavaMenu** are selected (meaning that identification properties are not captured for these objects).

Note: If you record on a specific test object, its identification properties will be captured even if the **Disable capture of the following objects** option is selected.

SAP GUI for Windows Applications

The following **Capture level** options are available for SAP GUI for Windows applications:

➤ **Complete.** Instructs QuickTest to save the property values of all objects in the application's open window/dialog box in the Active Screen of each step.

This option makes it possible for you to insert checkpoints and perform other operations on any object in the window/dialog box from the Active Screen of any step. However, it may result in longer recording times and require more disk space.

Note: The properties for inner objects of some container objects (such as table cells or tree nodes) are not captured in the Active Screen. Use the appropriate SAPGuiTable or SAPGuiTree methods to access information for these objects. For more information, see the **SAP GUI for Windows** section of the *HP QuickTest Professional Object Model Reference*.

➤ **Partial.** (Default) Instructs QuickTest to save properties of the recorded object and of its parent in the Active Screen of each step.

This option enables speedy recording and requires relatively little disk space. However, you can insert checkpoints and perform other operations only on the recorded object and on the window/dialog box itself. You cannot perform operations on the other objects displayed in the Active Screen.

➤ **None.** Disables the capture of Active Screen files for SAP GUI for Windows applications.

This option allows extremely fast recording and requires only minimum disk space. However, you cannot perform post-recording test editing (such as inserting checkpoints, output values, and so forth) from the Active Screen.

Notes:

- The property values of the objects in the Active Screen reflect the values at the time that the steps are added to your test (when information is sent to the SAP server). These values may potentially be different from the property values at the time that a particular step is performed.
- The Active Screen captures only the visible part of the SAP GUI for Windows applications window at the time that the step is added to the test.

Oracle Applications

The following **Capture level** options are available for Oracle applications:

- **Complete.** Instructs QuickTest to save all identification properties of all objects in the application's open window/dialog box in the Active Screen of each step.
- **Partial.** (Default) Instructs QuickTest to save all identification properties of all objects in the application's open window/dialog box in the Active Screen of the first step performed in that window, plus all identification properties of the recorded object only, in subsequent steps in the same window.
- **Minimum.** Instructs QuickTest to save all identification properties for the recorded object plus all identification properties for the parent objects in the recording hierarchy.
- **None.** Disables capture of Active Screen files for Oracle applications.

Windows Applications

The following **Capture level** options are available for Windows applications.

- **Complete.** Instructs QuickTest to save all properties of all objects in the application's open window/dialog box in the Active Screen of each step.

  This option makes it possible for you to insert checkpoints and perform other operations on any object in the window/dialog box, from the Active Screen for any step.

- **Partial** (Default)**.** Instructs QuickTest to save all properties of all objects in the application's open window/dialog box in the Active Screen of the first step performed in an application's window, plus all properties of the recorded object in subsequent steps in the same window.

  This option makes it possible for you to insert checkpoints and perform other operations on any object displayed in the Active Screen, while conserving recording time and disk space. Note that with this option the Active Screen information may not be fully updated for subsequent steps.

- **Minimum.** Instructs QuickTest to save properties only for the recorded object and its parent in the Active Screen of each step.

  This option enables speedy recording and requires relatively little disk space. However, you can insert checkpoints and perform other operations only on the recorded object and on the window/dialog box itself. You cannot perform operations on the other objects displayed in the Active Screen.

- **None.** Disables capture of Active Screen files for Windows applications.

  This option allows extremely fast recording and requires only a minimum of disk space. However, you cannot perform post-recording test editing from the Active Screen.

Terminal Emulator Applications

The following **Capture level** options are available for applications run on terminal emulators:

- **Complete.** Instructs QuickTest to save all properties of all objects in the application's open window/dialog box in the Active Screen of each step.

  This option makes it possible for you to insert checkpoints and perform other operations on any object in the window/dialog box, from the Active Screen for any step.

- **None.** Disables capture of Active Screen files for Terminal Emulator applications.

Web Options

You can specify whether QuickTest captures Web pages for the Active Screen.

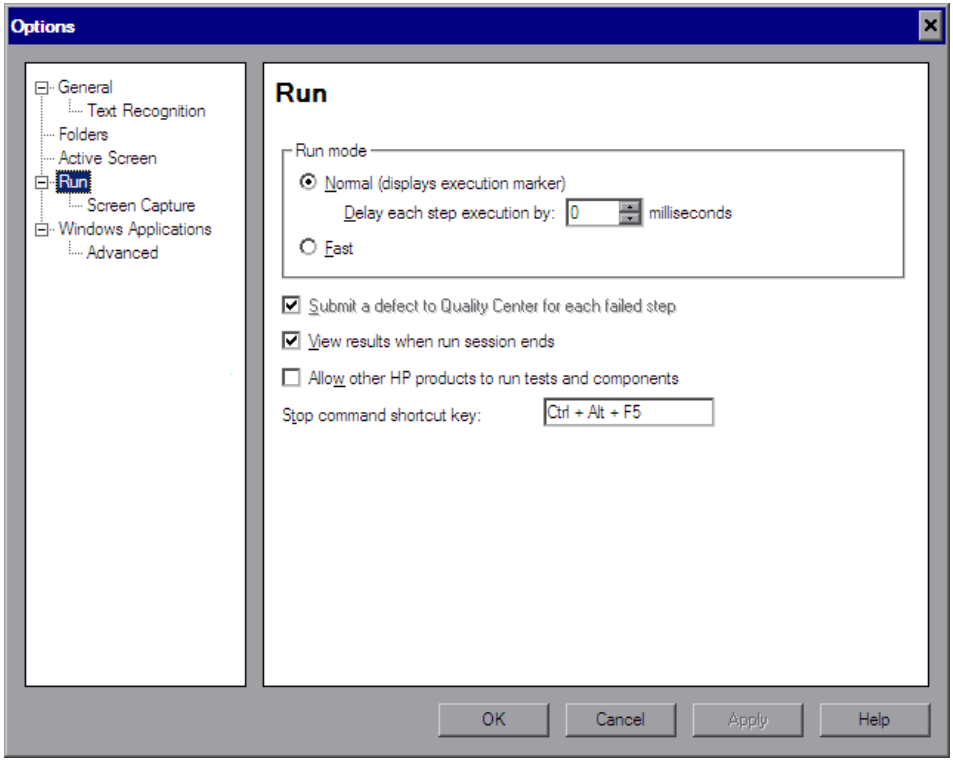
- **Disable Active Screen capture.** Disables the screen capture of all steps in the Active Screen.

  If you do not select this option, you can also delete Active Screen information after you have finished editing your test by selecting **Save As**, and clearing the **Save Active Screen files** check box. For more information, see "Saving a Test" on page 324.

- **Capture original HTML source.** Captures the HTML source of Web pages as they appear originally, before any scripts are run. Deselecting this option instructs QuickTest to capture the HTML source of Web pages after any dynamic changes have been made to the HTML source (for example, by scripts running automatically when the page is loaded).

Setting Run Testing Options

The Run pane options affect how QuickTest runs tests.



The Run node also contains the Screen Capture node. For more information, see “The Options Dialog Box: Run > Screen Capture Pane” on page 1255.

The Run pane includes the following options:

Option	Description
Run mode	Instructs QuickTest how to run your test: ➤ **Normal (displays execution marker).** Runs your test with the execution arrow to the left of the Keyword View or Expert View, marking each step or statement as it is performed. If the test contains multiple actions, the tree in the Keyword View **Item** column expands to display the steps, and the Expert View displays the script, of the currently running action. **Delay each step execution by.** You can specify the time in milliseconds that QuickTest should wait before running each consecutive step (up to a maximum of 10000 ms.) The **Normal** run mode option requires more system resources than the **Fast** option, described below. **Note:** You must have Microsoft Script Debugger installed to enable this mode. For more information, see the *HP QuickTest Professional Installation Guide*. ➤ **Fast.** Runs your test without the execution arrow to the left of the Keyword View or Expert View and does not expand the item tree or display the script of each action as it runs. This option requires fewer system resources. **Note:** When running a test set from Quality Center, tests are automatically run in **Fast** mode, even if **Normal** mode is selected.
Submit a defect to Quality Center for each failed step	Instructs QuickTest to automatically submit a defect to Quality Center for each failed step in your test. This option is available only when you are connected to a Quality Center project. For more information, see "Automatically Submitting Defects to a Quality Center Project" on page 1015.
View results when run session ends	Instructs QuickTest to display the results automatically following the run session.

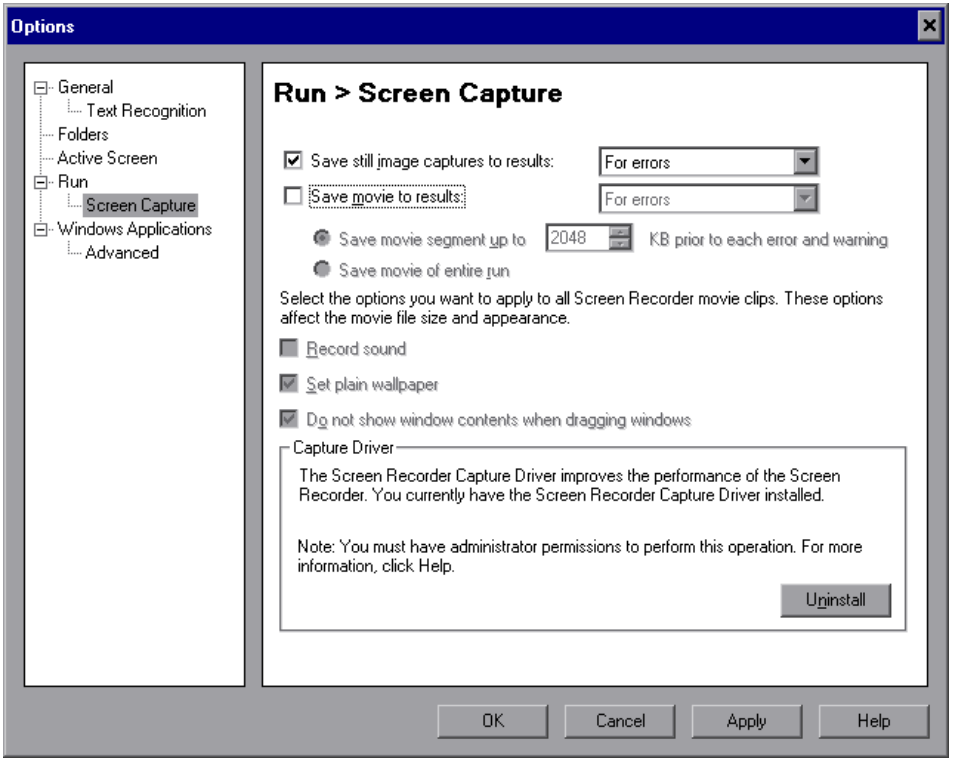
Option	Description
Allow other HP products to run tests and components	Enables other HP products such as Quality Center and Test Batch Runner to run QuickTest tests on this computer. **Note:** This option is not required to enable WinRunner to run QuickTest tests.
Stop command shortcut key	Enables you to define a shortcut key or key combination that stops the current QuickTest record or run operation, even if QuickTest is not in focus or is in hidden mode. Click in the field and then press the required key or key combination on the keyboard. The default key combination is **Ctrl+Alt+F5**. **Note:** It is important to define a shortcut that is not already defined for some other operation by the application being tested. If this is the case and: ➤ you open the application manually before you click **Record** or **Run**, the shortcut defined in the application will apply for its original purpose. ➤ you start a record or run session and QuickTest opens the application for you, the shortcut you define in the Run pane will stop the session.

The Options Dialog Box: Run > Screen Capture Pane

Description	Enables you to control when and how QuickTest captures screens of the application being tested.
How to Access	➤ Select the **Tools** > **Options** menu command and select the Run > Screen Capture node. ➤ Click the **Options** toolbar button  and select the Run > Screen Capture node.

Important Information	**Note for Vista users:** In addition to the options described below, if your Vista Windows color scheme is set to **Aero**, QuickTest automatically sets it to **Vista Basic** while capturing movies of a run session to maximize performance. The color scheme is returned to its previous settings when the run session ends.
Learn More	**Additional related topics**: "Additional References" on page 1259

Below is an image of the Run > Screen Capture pane in the Options dialog box:



Options Dialog Box: Run > Screen Capture Pane Options

Option	Description
Save still image captures to results	Instructs QuickTest when to capture still images of the application during the run session and save them in the test results. When images are available in the test results, QuickTest displays them in the bottom pane of the Result Details tab in the Test Results window. Clear the check box to disable this option, or select an option from the list: ➤ **Always.** Captures images for all steps in the run. ➤ **For errors.** Captures images only for failed steps. This is the default setting. ➤ **For errors and warnings.** Captures images only for steps that return a failed or warning status. For more information, see "Viewing Still Images and Movies of Your Application" on page 992. **Note:** This setting also affects the availability of other information displayed in the bottom pane of the test result details, such as: ➤ XML checkpoint and output value result details ➤ Bitmap checkpoint images (expected, actual, and difference)

Option	Description
Save movie to results	Instructs QuickTest when to capture a movie of the application during the run session and save it in the run results. When movies are available in the run results, QuickTest displays them in the Screen Recorder tab in the Test Results window. This option is disabled by default. Select the check box to enable this option and then select an option from the list: ➤ **Always.** Captures a movie of all steps in the run. ➤ **For errors.** Captures movies only for failed steps. ➤ **For errors and warnings.** Captures movies only for steps that return a failed or warning status. For more information, see "Viewing Still Images and Movies of Your Application" on page 992.
The following options are enabled only when the **Save movie to results** check box is selected.	
Save movie segment up to __ KB prior to each error and warning (Enabled only when **For errors** or **For errors and warnings** is selected in the **Save movie to results** option.)	When selected, QuickTest saves movie segments for each error (or warning). Each segment contains the specified number of kilobytes of the movie prior to the failed (or warning) step. You can enter any value from 400 (0.4 MB) to 2097152 (2 GB). If more than one segment is captured for a run session, QuickTest stores a single movie with that is comprised of all the relevant movie segments.
Save movie of entire run (Enabled only when **For errors** or **For errors and warnings** is selected in the **Save movie to results** option.)	When selected, QuickTest saves a movie of the entire run if at least one error (or warning) occurs.
Record sound	Instructs QuickTest to save sound with the movie of your application.
Set plain wallpaper	Sets the wallpaper of your desktop to a solid blue color for the duration of the run session.

Option	Description
Do not show window contents when dragging windows	Instructs Windows to display only the outline of a window, without its contents, whenever the window is dragged during the run session.
Capture Driver area	
Install/Uninstall button	Installs or uninstalls the Screen Recorder Capture Driver. The Screen Recorder Capture Driver improves the performance of the Screen Recorder during movie recording. **Note:** The Screen Recorder Capture Driver cannot be installed or uninstalled when running QuickTest via a remote connection.

Additional References

Related User Interface Topics	"Viewing Still Images and Movies of Your Application" on page 992

45

Setting Options for Individual Tests

You can control how QuickTest works with different tests by setting specific testing options for any individual test.

This chapter includes:

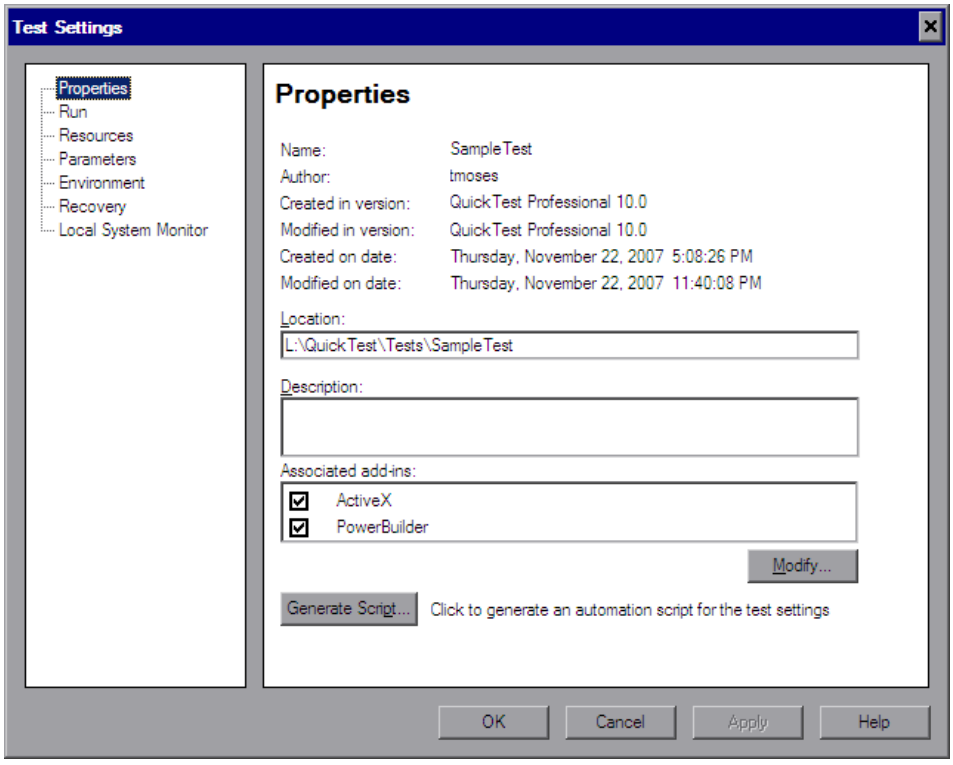
Using the Test Settings Dialog Box

You can use the Test Settings dialog box to set testing options that affect how QuickTest works with a specific test. For example, you can instruct QuickTest to run a parameterized test for only certain lines in the Data Table. The individual testing options that you specify are saved when you save the test.

Note: You can also set testing options that affect all tests. For more information, see Chapter 44, "Setting Global Testing Options."

To set testing options for an individual test:

1 Select **File** > **Settings** or click the **Settings** toolbar button. The Test Settings dialog box opens. It is divided into two parts: a navigation pane on the left and a settings display pane on the right.



2 Select the required node from the navigation tree and set the options in the settings display pane as necessary. See the table below for more information on the available options in each node.

3 Click **Apply** to apply your changes and keep the dialog box open, or click **OK** to save your changes and close the dialog box.

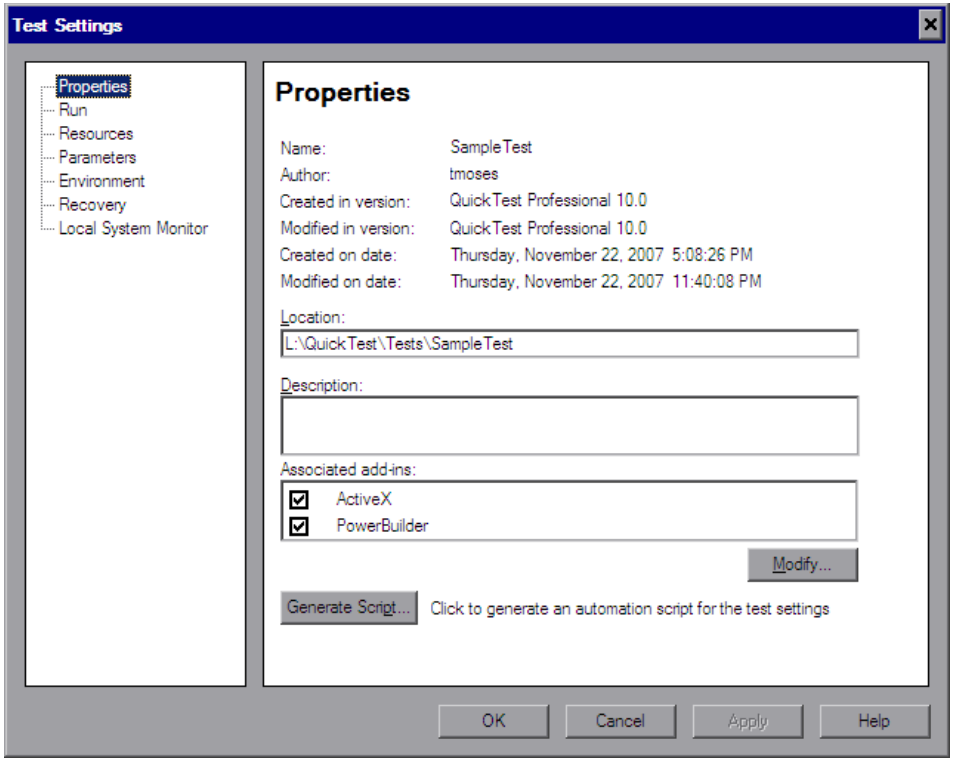
The navigation tree contains the following nodes:

Node	Options
Properties	Options for setting the properties of the test, for example, its description and associated add-ins. For more information, see "Defining Properties for Your Test" on page 1265.
Run	Options for setting the run session preferences. For more information, see "Defining Run Settings for Your Test" on page 1270.
Resources	Options for specifying resources you want to associate with your test, such as function libraries stored in VBScript function libraries. For more information, see "Defining Resource Settings for Your Test" on page 1274.
Parameters	Options for specifying input and output parameters for your test. For more information, see "Defining Parameters for Your Test" on page 1280.
Environment	Options for viewing existing built-in and user-defined environment variables, adding, modifying and saving user-defined environment variables, and selecting the active external environment variables file. For more information, see "Defining Environment Settings for Your Test" on page 1283.
Recovery	Options for setting how QuickTest recovers from unexpected events and errors that occur in your testing environment during a run session. For more information, see "Defining Recovery Scenario Settings for Your Test" on page 1291.
Local System Monitor	Options for activating and setting preferences for tracking system counters during a run session. For more information, see "Enabling System Monitoring for Your Test" on page 1296.

In addition to these nodes, the Test Settings dialog box may contain other nodes corresponding to any QuickTest add-ins that are installed or loaded. For more information on add-ins, see the relevant section in the *HP QuickTest Professional Add-ins Guide*.

Defining Properties for Your Test

You can use the Properties pane of the Test Settings dialog box (**File > Settings > Properties** node) to view and define general information about your test, including the add-ins associated with it. You can also choose to generate an automation script for the test settings.



The Properties pane of the Test Settings dialog box includes the following options:

Option	Description
Name	Indicates the name of the test. If the test is saved in a version-controlled project in Quality Center, the version number is also shown.
Author	Indicates the Windows user name of the person who created the test.
Created in version	Indicates the version of QuickTest used to create the test.
Modified in version	Indicates the version of QuickTest last used to modify the test.
Created on date	Indicates the date and time that the test was created.
Modified on date	Indicates the date and time that the test was last modified.
Location	Indicates the path and filename of the test.
Description	Enables you to specify a description for your test.
Associated add-ins	Displays the add-ins associated with the test. For more information, see "Associating Add-ins with Your Test" on page 1267.
Modify	Enables you to select the add-ins to associate with your test. For more information, see "Modifying Associated Add-Ins" on page 1268.
Generate Script	Generates an automation script containing the current test settings. For more information, see "Automating QuickTest Operations" on page 1403 or the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

Associating Add-ins with Your Test

When you open QuickTest, you select the add-ins to load from the Add-in Manager dialog box. You can create and edit tests that work with any environment for which the necessary add-in is loaded.

When you create a new test, the add-ins that are currently loaded are automatically associated with your test.

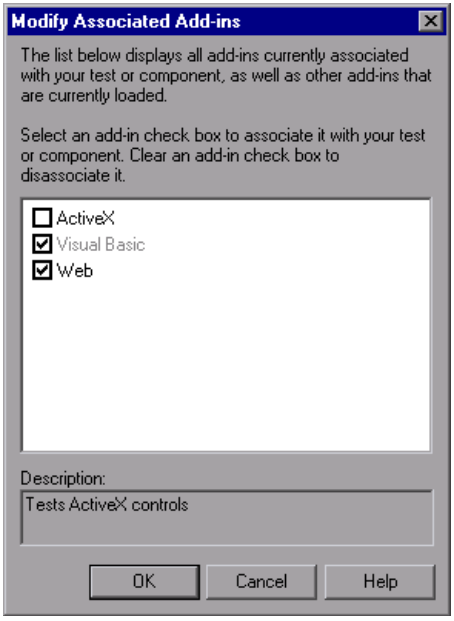
Choosing to associate an add-in with your test instructs QuickTest to check that the associated add-in is loaded each time you open that test.

When you open a test, QuickTest notifies you if an associated add-in is not currently loaded, or if you have loaded add-ins that are not currently associated with your test. This process ensures that your run session will not fail due to unloaded add-ins and reminds you to add required add-ins to the associated add-ins list if you plan to use them with the currently open test. For more information on loading and working with add-ins, see the *HP QuickTest Professional Add-ins Guide*.

Quality Center uses the associated add-ins list to determine which add-ins to load when it opens QuickTest to run or view a test. For more information on working with Quality Center, see Chapter 51, "Integrating with Quality Center."

Modifying Associated Add-Ins

You can associate or disassociate add-ins with your test in the Modify Associated Add-ins dialog box.



This dialog box lists all the add-ins currently associated with your test, as well as any other add-ins that are currently loaded in QuickTest. Add-ins that are associated with your test but not currently loaded are shown dimmed.

Note: This list might also include child nodes representing add-ins that you or a third party developed to support additional environments or controls using add-in extensibility. For more information, see the relevant Add-in Extensibility Developer's Guide (available with the extensibility setup).

You can select the check boxes for add-ins that you want to associate with your test, or clear the check boxes for add-ins that you do not want to associate with your test. If the Modify Associated Add-ins dialog box contains a child add-in, and you select it, the parent add-in is selected automatically. If you clear the check box for a parent add-in, the check boxes for its children are also cleared.

In the above example:

- Web is loaded and associated with the test.
- ActiveX is loaded, but not associated with the test.
- Visual Basic is associated with the test, but is not loaded.

Note: If a specific add-in is not currently loaded, but you want to associate it with your test, reopen QuickTest and load the add-in from the Add-in Manager. If the Add-in Manager dialog box is not displayed when you open QuickTest, you can choose to display it the next time you open QuickTest. To do so, select **Display Add-in Manager on startup** from the General pane of the Options dialog box.

For more information on the Options dialog box, see Chapter 44, "Setting Global Testing Options."

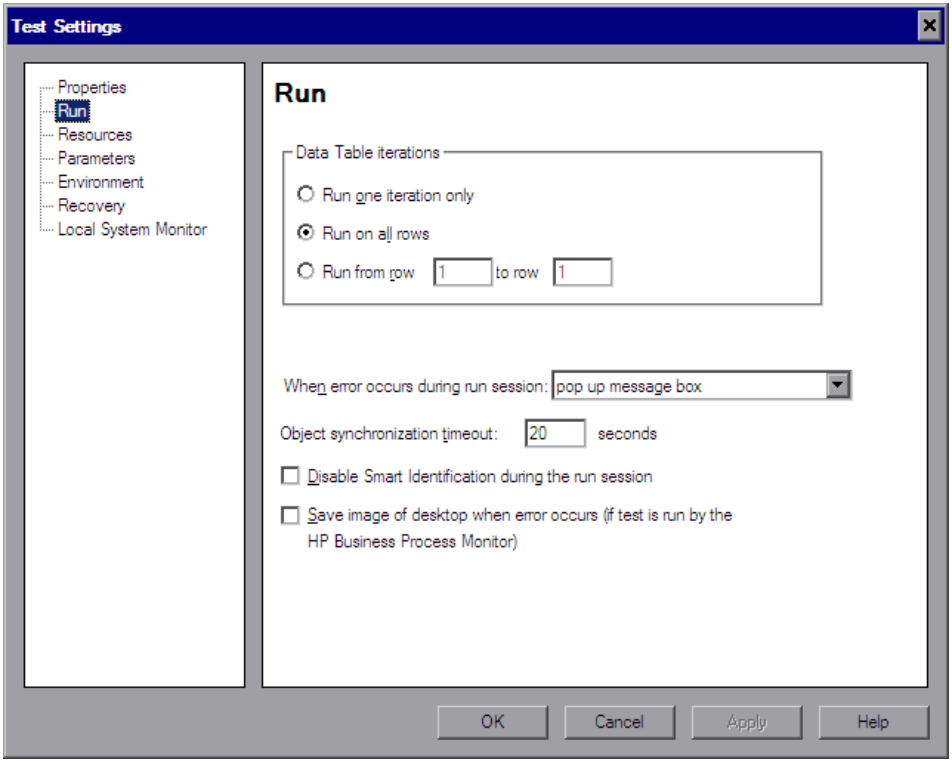
For more information on the Add-in Manager, see the section on working with QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.

You can also retrieve this list and load add-ins accordingly using an automation script. For more information on working with automation scripts, see the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

Defining Run Settings for Your Test

When you run a test, QuickTest performs the test steps on your application.

You can use the Run pane in the Test Settings dialog box (**File** > **Settings** > **Run** node) to choose what to do when an error occurs during the run session, set the object synchronization timeout and choose whether or not to disable the Smart Identification mechanism for the test.



By default, when you run a test with global Data Table parameters, QuickTest runs the test for each row in the Data Table, using the parameters you specified. For more information, see “Choosing Global or Action Data Table Parameters” on page 643.

You can use the Run pane to instruct QuickTest to run iterations on a test only for certain lines in the Global tab in the Data Table.

Note: The Run pane of the Test Settings dialog box applies to the entire test. You can set the run properties for an individual action in a test from the Run tab in the Action Call Properties dialog box of a selected action. For more information on action run properties, see "Setting the Run Properties for an Action" on page 482.

The Run pane includes the following options:

Option	Description
Data Table iterations	Specifies the iterations for the test. Select an option: ➤ **Run one iteration only.** Runs the test only once, using only the first row in the global Data Table. ➤ **Run on all rows.** Runs the test with iterations using all rows in the global Data Table. ➤ **Run from row __ to row __.** Runs the test with iterations using the values in the global Data Table for the specified row range.
When error occurs during run session	Specifies how QuickTest responds to an error during the run session. For more information, see "Specifying the Response to an Error" on page 1272.
Object synchronization timeout	Sets the maximum time (in seconds) that QuickTest waits for an object to load before running a step in the test. **Note:** When working with Web objects, QuickTest waits up to the amount of time set for the **Browser navigation timeout** option, plus the time set for the object synchronization timeout. For more information on the **Browser navigation timeout** option, see the *HP QuickTest Professional Add-ins Guide*.

Option	Description
Disable Smart Identification during the run session	Instructs QuickTest not to use the Smart Identification mechanism during the run session. **Note:** When you select this option, the **Enable Smart Identification** check boxes in the Object Properties and Object Repository dialog boxes are disabled, although the settings are saved. When you clear this option, the **Enable Smart Identification** check boxes return to their previous on or off setting.
Save image of desktop when error occurs (if test is run by the HP Business Process Monitor)	This option is applicable only to tests that are run by the Business Process Monitor component of HP Business Availability Center. Selecting this option instructs QuickTest to capture a snapshot of the desktop if an error occurs during a run session of a test initiated by the Business Process Monitor. The image is saved in Business Availability Center. The Business Process Monitor forwards the run results to the Business Availability Center servers.

Specifying the Response to an Error

By default, if an error occurs during the run session, QuickTest displays a popup message box describing the error. You must click a button on this message box to continue or end the run session.

You can accept the **popup message box** option or you can specify a different response by choosing one of the alternative options in the list in the **When error occurs during run session** box:

- **proceed to next action iteration**. QuickTest proceeds to the next action iteration when an error occurs.
- **stop run**. QuickTest stops the run session when an error occurs.
- **proceed to next step**. QuickTest proceeds to the next step in the test when an error occurs.

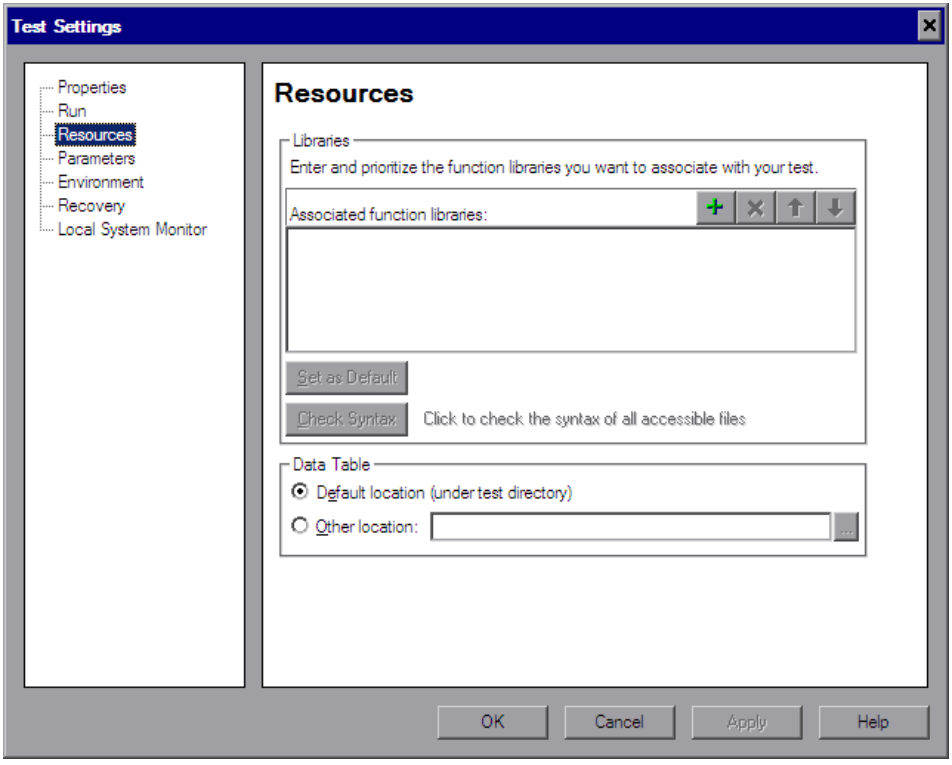
QuickTest first performs any recovery scenarios associated with the test, and performs the option selected above only if the associated recovery scenarios do not resolve the error. For more information, see "Defining Recovery Scenario Settings for Your Test" on page 1291.

Note: If you are working with many tests, you may want to use a QuickTest automation script to set a different value for each test. To access the automation script line that controls this option, you can use the **Generate Script** button in the Properties pane of the Test Settings dialog box.

For more information, see "Automating QuickTest Operations" on page 1403 or the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

Defining Resource Settings for Your Test

You can use the Resources pane of the Test Settings dialog box (**File > Settings > Resources** node) to associate specific files with your test, such as VBScript function libraries and Data Table files. You can also set the currently associated function library settings as the default settings for all new tests.



Note: Object repositories are associated with individual action(s) in your test. You can associate an object repository with an action using the Action Properties dialog box (**Edit > Action > Action Properties**) and the Associate Repositories dialog box (**Resources > Associate Repositories**).

The Resources pane in the Test Settings dialog box includes the following option areas:

Option Area	Description
Libraries	Displays the list of function libraries associated with your test. You can add, delete, and prioritize the files. You can also set the default function libraries for new tests. For more information, see "Specifying Associated Function Libraries" on page 1276.
Set as Default	Sets the current list of function libraries as the default list to be associated with new tests. **Note:** The **Set as Default** option is available for tests only. It is enabled when the setting for this test is different from the default for all tests. **Caution:** If the default function library is moved or renamed, QuickTest will not be able to locate it. The function library will be displayed in the Missing Resources pane when new actions or tests are created. For information on resolving missing resources, see Chapter 41, "Handling Missing Resources."
Check Syntax	Verifies whether any of the associated function libraries contain syntax errors that will prevent the test from running properly. Click the **Check Syntax** button to check the files for syntax errors before finalizing the test. If any syntax errors are found, the Information pane opens listing the files containing syntax errors. Otherwise, an information box opens confirming that the syntax in all of the function libraries is valid. **Note:** QuickTest checks only the associated function libraries that can be accessed. For example, if an associated function library is stored in a Quality Center project to which you are not currently connected, its syntax will not be checked.

Option Area	Description
Data Table	Specifies the location of the Data Table to be used in your test: ➤ **Default location (under test directory).** Instructs QuickTest to use data stored in the default Data Table location under the test folder. ➤ **Other location.** Instructs QuickTest to use data stored in the specified Data Table location. The Data Table can be any Microsoft Excel (**.xls**) file. For more information on Data Tables, see "About Working with Data Tables" on page 1197. **Note:** You can specify Microsoft Excel files stored in Quality Center as Data Tables. For more information, "Using Data Table Files with Quality Center" on page 1212.

Specifying Associated Function Libraries

The **Associated function libraries** area of the Resources pane indicates the list of function libraries associated with your test. QuickTest searches these files for the VBScript functions, subroutines, and so forth that are specified in the test.

The order of the function libraries in the list determines the order in which QuickTest searches for a function or subroutine that is called from a step in your test. If there are two functions or subroutines with the same name, QuickTest uses the first one it finds. For more information, see "Working with Associated Function Libraries" on page 919.

You can enter an associated function library using an absolute or relative path. If you enter it as a relative path, then during a run session, QuickTest searches for the file in the directory for the current test, and then in the folders listed in the Folders pane of the Options dialog box. For more information, see "Setting Folder Testing Options" on page 1237 and "Using Relative Paths in QuickTest" on page 316.

Notes:

➤ When working with tests, if your function libraries are stored in the file system and you want other users or HP products to be able to run this test on other computers, you can set the file path as a relative path (click the path once to highlight it, and then click it again to enter edit mode). Any users who want to run this test should then specify the drive letter and folder in which QuickTest should search for the relative path in the Folders pane of the Options dialog box (**Tools** > **Options** > **Folders** node). For more information, see "Setting Folder Testing Options" on page 1237, and "Using Relative Paths in QuickTest" on page 316.

Important: If you are working with the Resources and Dependencies model with Quality Center 10.00, you should store your function libraries in the Quality Center Test Resources module and specify an absolute Quality Center path in the Folders pane. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.

➤ You can also add, delete and prioritize the function libraries associated with your test using the Resources pane. For more information, see "The Resources Pane" on page 1161.

You can add, delete and prioritize the function libraries associated with your test using the function library control buttons:

Option	Description
	Associates a function library with the test. You can enter the absolute or relative path and filename of the function library, or use the browse button to locate the required file. If the function library contains syntax errors, a message opens stating that your test will fail because of these syntax errors. The function library can be located in the file system or in a Quality Center project folder. For more information on associating a function library stored in Quality Center, see “Associating Function Libraries in Quality Center Project Folders” on page 1279, below. **Note**: If you are working with the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see “Considerations for Working with Relative Paths in Quality Center” on page 1450.
	Removes an associated function library from the list.
	Assigns a higher priority to the selected function library.
	Assigns a lower priority to the selected function library.

Associating Function Libraries in Quality Center Project Folders

When you are connected to Quality Center and you click the  button, QuickTest adds [QualityCenter], and displays a browse button so that you can locate the Quality Center path.

When not connected to Quality Center, you can add a file located in a Quality Center project folder by holding the SHIFT key and clicking the  button. QuickTest adds [QualityCenter], and you can enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [QualityCenter]. For example: [QualityCenter] Subject\Tests.

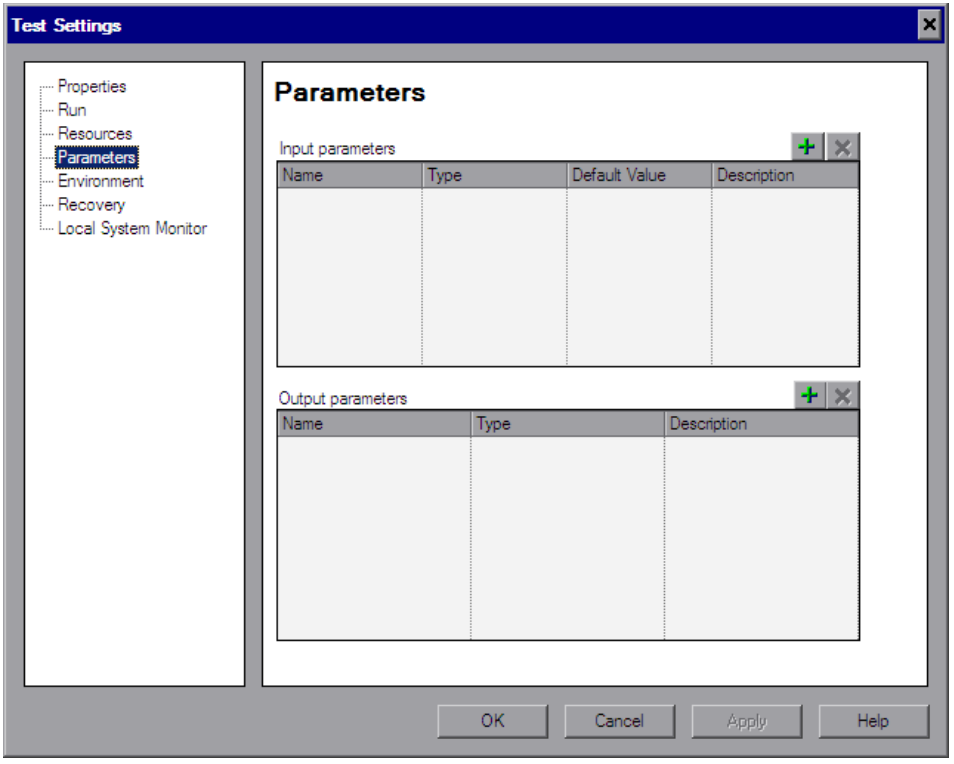
Note: When running a test, QuickTest uses associated function libraries from Quality Center project folders only when you are connected to the corresponding Quality Center project.

For more information on working with Quality Center projects, see Chapter 51, "Integrating with Quality Center."

Defining Parameters for Your Test

You use the Parameters pane of the Test Settings dialog box (**File** > **Settings** > **Parameters** node) to define input parameters that pass values into your test and output parameters that pass values from your test to external sources. You can also use the Parameters pane to modify or delete existing test parameters.

Test parameters are similar to Action parameters. For information on Action parameters, see "Setting Action Parameters" on page 472.



The Parameters pane contains the following parameter lists:

- **Input parameters**. Specifies the parameters that the test can receive values from the source that runs or calls it.
- **Output parameters**. Specifies the parameters that the test can pass to the source that runs or calls it.

You can edit an existing parameter by selecting it in the appropriate list and modifying its details.

You can add and remove input and output parameters for your test using the parameter control buttons:

Option	Description
	Adds a parameter to the appropriate parameter list. Enter a name for the new parameter (case sensitive) and select the parameter type. You can enter a description for the parameter, for example, the purpose of the parameter in the test. If you are defining an input parameter, a default value for the specified parameter type is automatically entered. You can modify a default value for the parameter in the **Default Value** column. For more information, see “Defining Default Values for Input Parameters” on page 1282, below. You define test parameters in the same way you define action parameters. For information on defining parameters and parameter types, see “Setting Action Parameters” on page 472.
	Removes the selected parameter from the test.

Defining Default Values for Input Parameters

When a test runs, the actual values used for parameters are generally those sent by the application calling the test (either QuickTest or Quality Center) as described in the table below:

Document Type:	Called From:	Parameter Values Specified In:
Test	**QuickTest**	Input Parameters tab of the Run dialog box. For more information, see "Running Your Entire Test" on page 955.
Test	**Quality Center**	Test Run Properties dialog box (Test Lab module). For more information, see the *HP Quality Center User Guide*.

If, when a test runs, a value is not supplied by QuickTest or Quality Center for one or more input parameters, QuickTest uses the default value for the parameter.

When you define a new parameter in the Parameters pane of the Test Settings dialog box, you can specify the default value for the parameter or you can keep the default value that QuickTest assigns for the specified parameter type as follows:

Value Type	QuickTest Default Value
String	Empty string
Boolean	True
Date	The current date
Number	0
Password	Empty string
Any	Empty string

Using Test Parameters in Steps

You can directly access test parameters only when parameterizing the value of a top-level action input parameter or when specifying the storage location for a top-level output parameter. To use values supplied for test parameters in steps within an action, you must pass the test parameter to the action containing the step. For more information, see "Setting Action Parameters" on page 472.

Defining Environment Settings for Your Test

The Environment pane of the Test Settings dialog box (**File** > **Settings** > **Environment** node) displays existing built-in and user-defined environment variables. It also enables you to add, modify, or delete internal user-defined environment variables, save the defined variables to an external **.XML** file, and retrieve them from a file.

If you export your user-defined variables to an external **.XML** file, you can then use the exported environment variable file with any other test.

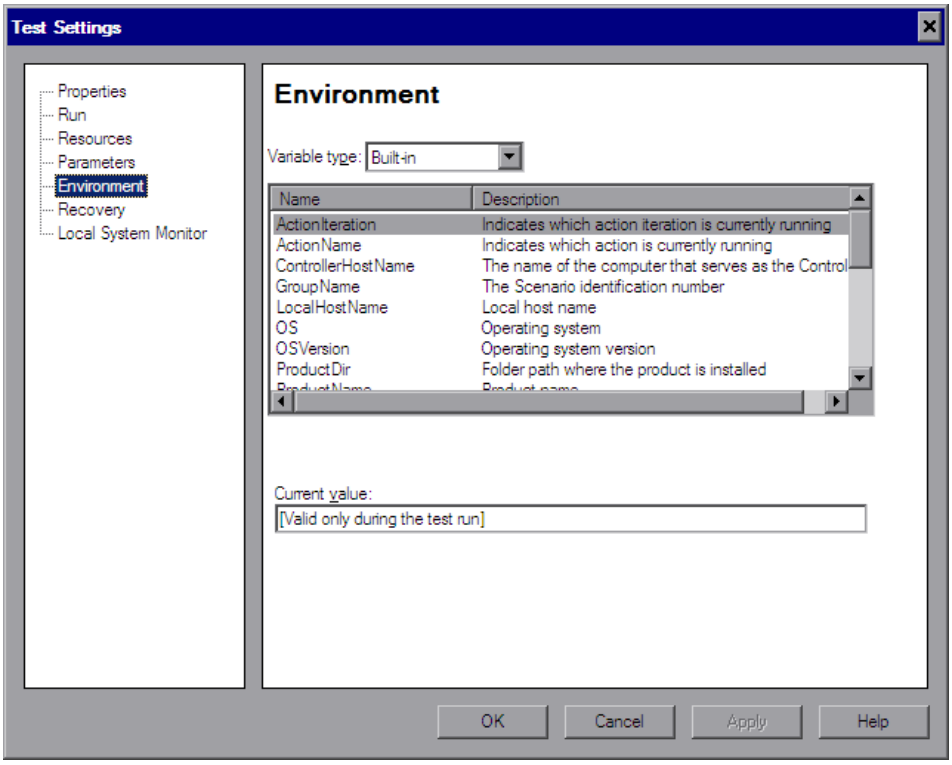
For more information on environment variables and environment parameters, see "Using Environment Variable Parameters" on page 645.

The Environment pane includes the following options for the **Variable type**:

- **Built-in.** Displays the built-in environment variables defined by QuickTest Professional and their current values.
- **User-defined.** Displays both internal and external user-defined environment variables and their current values.

Built-in Environment Variables

When **Built-in** is selected, the Environment pane lists the built-in environment variables defined by QuickTest Professional.

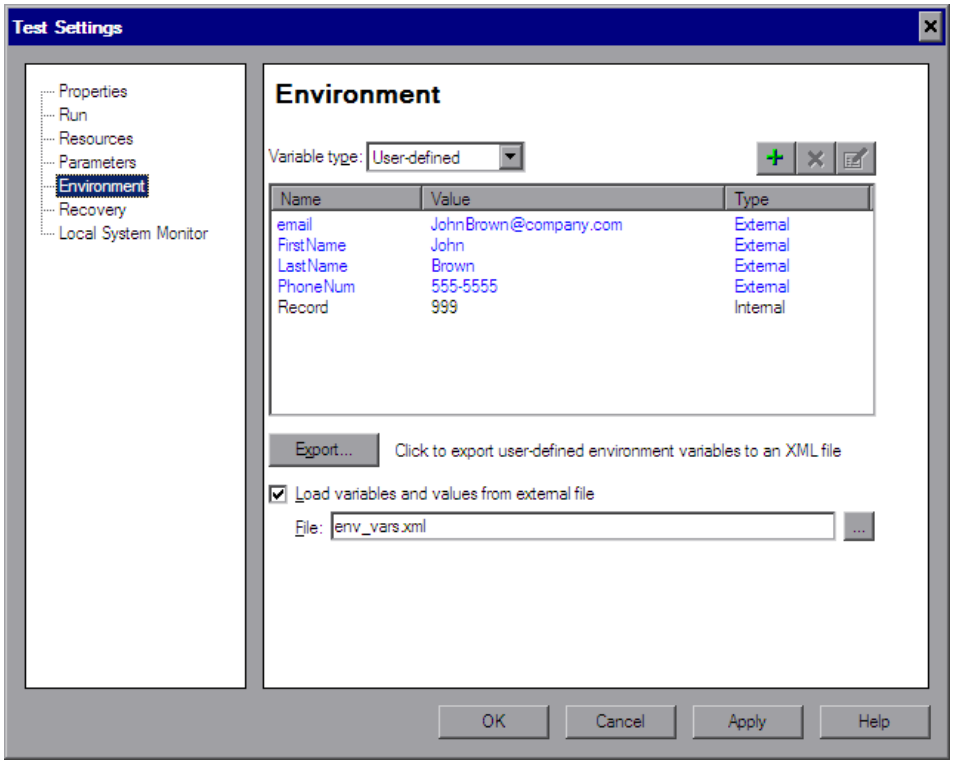


The following information is displayed for built-in environment variables:

- **Name.** The name of each built-in environment variable
- **Description.** A short description of each built-in environment variable
- **Current value.** The current value of the selected environment variable

User-Defined Environment Variables

When **User-defined** is selected, the Environment pane lists the user-defined environment variables available for the test.



Note: Variables from an external environment variables file are displayed in blue. Internal environment variables are displayed in black.

The Environment pane provides the following information for user-defined environment variables:

- **Name.** The name of each user-defined variable
- **Value.** The value assigned to each user-defined variable
- **Type.** The type of each user-defined variable: **Internal** or **External**. Internal environment variables are available only to the test in which they are defined.

The Environment pane provides the following options for user-defined environment variables:

Option	Description
	Enables you to define a new internal environment variable and add it to the list. For more information, see "Adding User-Defined Environment Variables", below.
	Deletes a selected internal environment variable from the list. **Note:** After you confirm the deletion of the environment variable, you cannot retrieve it, even if you click **Cancel** in the Test Settings dialog box.
	Enables you to edit the value of a selected internal environment variable or to view the properties of a selected external environment variable. For more information, see "Viewing and Modifying User-Defined Environment Variables" on page 1287.
Export	Exports your user-defined environment variables to an external **.XML** file for use with other tests. You can then use the exported environment variable file with any test. For more information, see "Exporting and Loading User-Defined Environment Variables" on page 1289.
Load variables and values from external file	Loads the variables saved in the **.XML** file that you specify for use with your test. For more information, see "Exporting and Loading User-Defined Environment Variables" on page 1289.

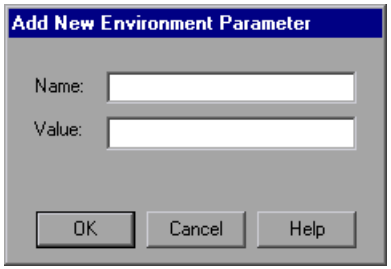
Adding User-Defined Environment Variables

You can add internal user-defined environment variables in the Environment pane of the Test Settings dialog box. Internal environment variables are available only to the test in which they are defined.

To add internal user-defined environment variables:

1 In the **Variable type** box of the Environment pane, select **User-defined**.

2 Click the **New** button. The Add New Environment Parameter dialog box opens.



3 Enter a definition for the variable as follows:

- **Name.** Enter the name of the variable.
- **Value.** Enter the value of the variable.

4 Click **OK** to save your changes and close the Add New Environment Parameter dialog box. The variable is added to the list (displayed in black) in the Environment pane of the Test Settings dialog box.

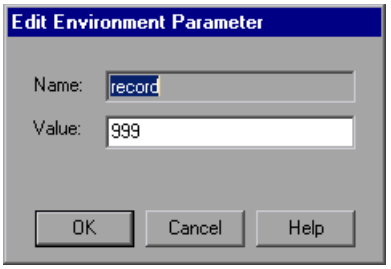
Viewing and Modifying User-Defined Environment Variables

You can edit the values of internal user-defined environment variables in the Environment pane of the Test Settings dialog box. You can also view the properties of external user-defined variables.

You can copy the values of internal and external variables for use in other areas of QuickTest, for example, in the Data Table.

To modify or copy an internal user-defined environment variable:

1 In the Environment pane of the Test Settings dialog box, double-click the internal variable, or select it and click the **View/Edit Environment Variable** button. The Edit Environment Parameter dialog box opens.



2 To modify the value of the variable, enter a different value in the **Value** box.

3 To copy the value of the variable to the Clipboard, select the value text, right-click, and select **Copy**.

4 Click **OK** to save your changes and close the Edit Environment Parameter dialog box. The value of the variable is updated in the Environment pane of the Test Settings dialog box.

To view an external user-defined environment variable:

1 In the Environment pane of the Test Settings dialog box, double-click the external variable you want to view, or select it and click the **View/Edit Environment Variable** button. The View Environment Parameter dialog box displays the details of the selected variable.

If the variable has a complex value (a value that cannot be displayed entirely in the **Value** box), you can click the **View/Edit Complex Value** button to view the contents of the value.

2 To copy the value of the variable to the Clipboard, select the value text, right-click and select **Copy**.

3 Click **Close** to close the View Environment Parameter dialog box.

Exporting and Loading User-Defined Environment Variables

You can export your user-defined environment variables to an external **.XML** file for use with other tests. You can then use the exported environment variables with any test, by loading them from the file as external user-defined environment variables.

If the file is saved to the file system, its values are loaded each time the test runs. If the file is saved to a Quality Center project, its values are loaded when the test is first loaded. If the values are changed after the test is loaded, the new values will not be used by QuickTest, until the next time the test is loaded.

To export user-defined environment variables:

1 In the Environment pane of the Test Settings dialog box, click the **Export** button. The Save Environment Variable File dialog box opens, enabling you to export the current list of user-defined variables and values to an **.XML** file.

2 In the sidebar, select the location in which you want to save the file, for example, File System or Quality Center Test Resources.

3 Browse to and select the folder in which you want to save the file.

4 In the **File name** box, enter a name for the file and click **Save**.

Tip: If you want to save the file as an attachment to a test in the Test Plan module in Quality Center, select **Quality Center Test Plan** in the sidebar, browse to and double-click the test, and then click **Save**.

Note: When you specify a path to a resource in the file system or in Quality Center 9.x, QuickTest checks if the path, or a part of the path, exists in the Folders pane of the Options dialog box (**Tools** > **Options** > **Folders** node). If the path exists, you are prompted to define the path using only the relative part of the path you entered. If the path does not exist, you are prompted to add the resource's location path to the Folders pane and define the path relatively. For more information, see "Using Relative Paths in QuickTest" on page 316.

If you are working with the Resources and Dependencies model with Quality Center 10.00, you should specify an absolute Quality Center path. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.

To load variables from an external user-defined environment variable file:

1. In the Test Settings dialog box navigation pane, click the **Environment** node.
2. In the Environment pane, select **User-defined** from the **Variable type** box. The options for user-defined variables are displayed.
3. Select the **Load variables and values from external file** check box.

4 In the **File** box, enter the file name or click the browse button to find the external user-defined variable file.

The environment variables loaded from the selected file are displayed in blue in the Environment pane of the Test Settings dialog box.

Note: If you enter a relative path for the environment variable file, QuickTest searches for the file in the folders listed in the Folders pane of the Options dialog box. For more information, see "Setting Folder Testing Options" on page 1237 and "Using Relative Paths in QuickTest" on page 316.

If you are working with the Resources and Dependencies model with Quality Center 10.00, you should specify an absolute Quality Center path. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.

For more information on built-in and user-defined variables, and for information on how to create an external user-defined environment variable file, see "Using Environment Variable Parameters" on page 645.

Defining Recovery Scenario Settings for Your Test

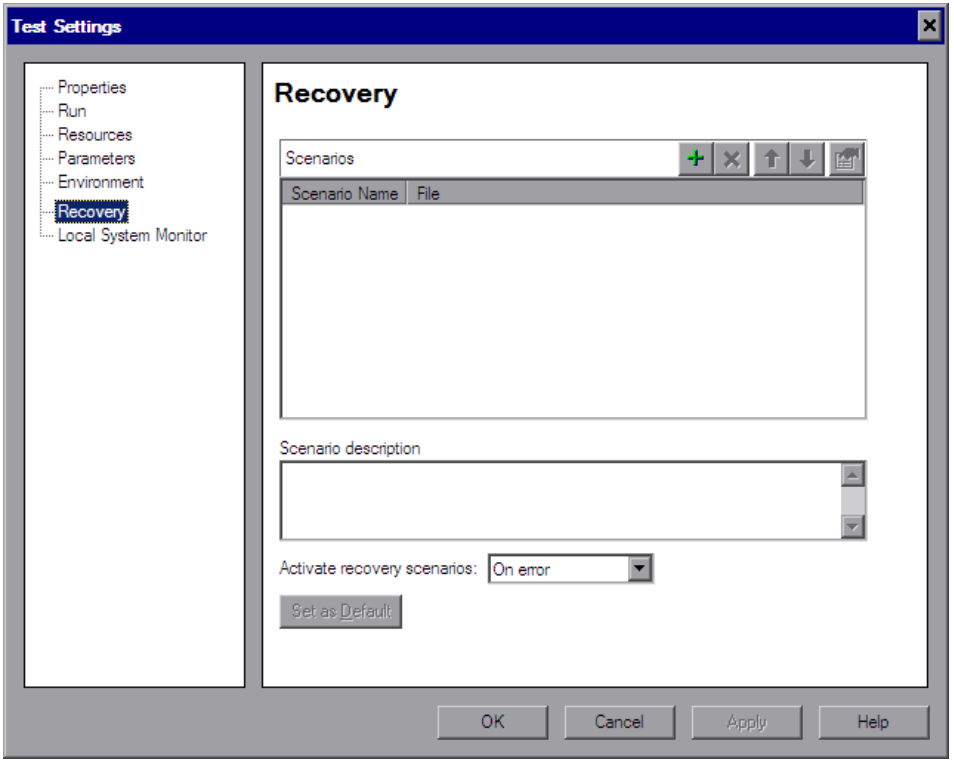
The Recovery pane of the Test Settings dialog box (**File** > **Settings** > **Recovery** node) displays a list of all recovery scenarios associated with the current test. It also enables you to associate additional recovery scenarios with the test, remove scenarios from the test, change the order in which they are applied to the run session, and view a read-only summary of each scenario.

You can enable or disable specific scenarios or the entire recovery mechanism for the test.

If you are working with tests, you can specify that the current list of scenarios be used as the default for all new tests.

You can also associate, remove, enable, disable, prioritize, and view the properties of the recovery scenarios associated with your test in the Resources pane. For more information, see “The Resources Pane” on page 1161.

For more information on recovery scenarios, see Chapter 48, “Defining and Using Recovery Scenarios.”



The Recovery pane includes the following option areas:

Option Area	Description
Scenarios	Displays the name and recovery file path for each recovery scenario associated with your test. You can add, delete, and prioritize the scenarios in the list, and you can edit the file path for a selected file. For more information, see Chapter 45, "Specifying Associated Recovery Scenarios", below.
Scenario description	Displays the textual description of the scenario selected in the **Scenarios** box.
Activate recovery scenarios	Instructs QuickTest to check whether to run the associated scenarios as follows: ➤ **On every step.** The recovery mechanism is activated after every step. ➤ **On error.** The recovery mechanism is activated only after steps that return an error return value. ➤ **Never.** The recovery mechanism is disabled. **Note:** Choosing **On every step** may result in slower performance during the run session.
Set as Default	Sets the current list of recovery scenario files as the default list to be associated with new tests. **Note:** The **Set as Default** option is available for tests only. It is enabled when the setting for this test is different from the default for all tests. **Caution:** If the file containing the recovery scenarios is moved or renamed, QuickTest will not be able to locate it. The recovery scenario file will be displayed in the Missing Resources pane when new actions or tests are created. For information on resolving missing resources, see Chapter 41, "Handling Missing Resources."

Note: When working with tests, if your recovery files are stored in the file system and you want other users or HP products to be able to run this test on other computers, you should set the recovery file path as a relative path (click the path once to highlight it, and then click it again to enter edit mode). Any users who want to run this test should then specify the drive letter and folder in which QuickTest should search for the relative path in the Folders pane of the Options dialog box (**Tools** > **Options** > **Folders** node). For more information, see "Setting Folder Testing Options" on page 1237 and "Using Relative Paths in QuickTest" on page 316.

If you are working with the Resources and Dependencies model with Quality Center 10.00, you should store your recovery files in the Quality Center Test Resources module and specify an absolute Quality Center path in the Folders pane. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.

Specifying Associated Recovery Scenarios

You can select or clear the check box next to each scenario to enable or disable it for the current test.

You can also edit the recovery scenario file path by clicking the path once to highlight it, and then clicking it again to enter edit mode. For example, you may want to modify an absolute file path to be a relative file path. If you modify a recovery scenario file path, ensure that the recovery scenario exists in the new path location before running your test.

Scenarios are indicated by the following icons:

Icon	Description
	Indicates that the recovery scenario is triggered by a specific pop-up window in an open application during the run session.
	Indicates that the recovery scenario is triggered when the property values of an object in an application match specified values.
	Indicates that the recovery scenario is triggered when a step in the test does not run successfully.
	Indicates that the recovery scenario is triggered when a specified application fails during the run session.
	Indicates that the recovery scenario is no longer available for the test—possibly because the recovery file has been renamed or moved, or can no longer be accessed by QuickTest. When an associated recovery file is not available during a run session, a message is displayed in the test results.

Note: The default recovery scenarios provided with QuickTest are installed in your QuickTest installation folder. The paths specifying the default recovery scenarios in the Recovery pane use an environment variable (**%ProductDir%**) in the file path. This enables QuickTest to locate these recovery scenarios when tests associated with them are run on different computers or by different HP products. Do not modify the file paths of these default recovery scenarios or attempt to use the environment variable for any other purpose.

You can add, delete, and prioritize the recovery scenario files associated with your test using the recovery scenario file control buttons:

Option	Description
	Opens the Add Recovery Scenario dialog box, which enables you to associate one or more recovery scenarios with the test. For more information, see "Adding Recovery Scenarios to Your Test" on page 1373.
	Removes the selected recovery scenario from the test.
	Moves the selected scenario up in the list, giving it a higher priority.
	Moves the selected scenario down in the list, giving it a lower priority.
	Displays summary properties for the selected recovery scenario in read-only format. For more information, see "Viewing Recovery Scenario Properties" on page 1376.

Enabling System Monitoring for Your Test

You can use the Local System Monitor pane of the Test Settings dialog box (**File** > **Settings** > **Local System Monitor** node) to activate and set preferences for tracking system counters during a run session.

The local system monitor tracking options enable you to track application performance counters during a run session. These counters enable you to monitor the resources used by your application.

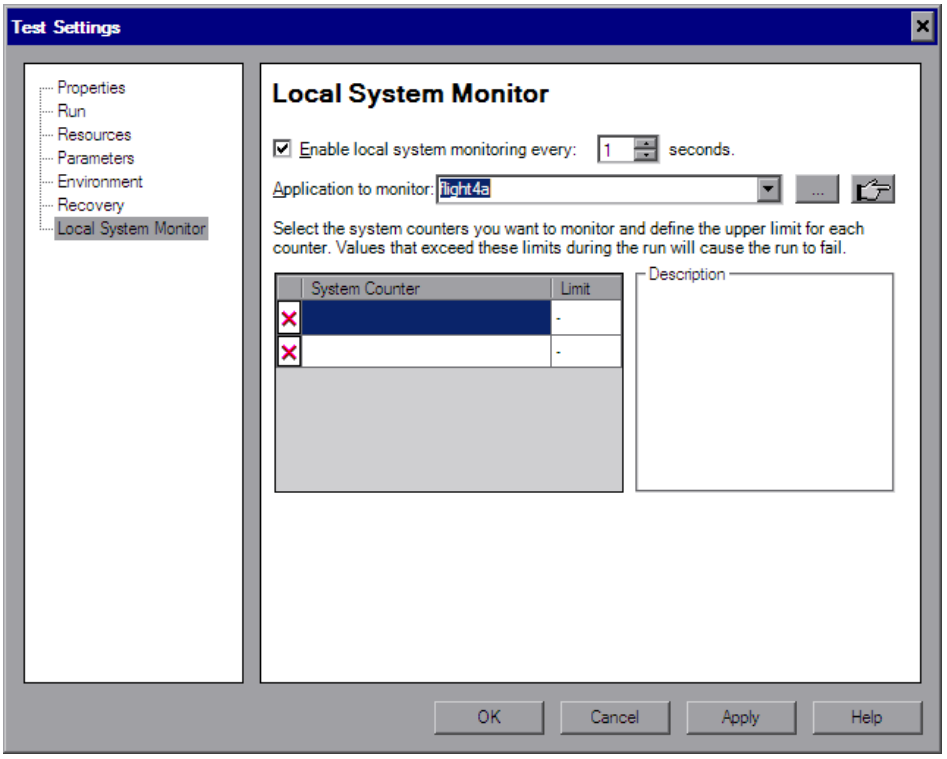
The system counters that can be monitored are the process counters accessible through the Performance Console (Select **Start** > **Run** > and then enter Perfmon). For information on the process counters accessible through the Performance Console, see the Performance Console Help.

You can also define limits for the counters. If the specified counters exceed these limits, the test run will fail. The results of the system counters are viewed in the Test Results window. For more information, see "Viewing System Monitor Results" on page 1063.

The Test Settings Dialog Box: Local System Monitor Pane

Description	Enables you to activate system monitoring, and define the system counters you want to track during a run session.
How to Access	**File** menu> **Settings** item > **Local System Monitor** node
Important Information	➤ The Local System Monitor data that is captured during a test run is displayed in the Test Results window. For more information, see "Viewing System Monitor Results" on page 1063. ➤ If there is more than one process with the same name running during a test, and you monitor a counter for that process (for example, you select to monitor a counter for the iexplorer.exe process, and more than one Internet Explorer browser is open on your desktop during the run session), the counter will be sampled from the application that contains at least one test object from the test. If more than one application meets this criterion, only one application will be monitored.
Learn More	**Conceptual overview**: "Enabling System Monitoring for Your Test" on page 1296.

Below is an image of the Local System Monitor pane in the Test Settings dialog box:



The Test Settings Dialog Box: Local System Monitor Pane Options

Option	Description
Enable local system monitoring every: __ seconds	Defines the frequency in seconds, by which the system counters for this application will be checked. Use the up and down arrows or enter a value in the edit box to change the number of seconds. The minimum value is one second.
Application to monitor	Defines the application whose system counters you want to monitor. You can define the application in one of the following ways: ➤ Enter the name of the application's executable file (without file extension) in the edit box. ➤ Click the down arrow in the edit box for a list of applications previously run in QuickTest, currently running applications, and applications currently specified in the Windows Applications tab of the Record and Run Settings dialog. ➤ Click the browse button ... and browse to the application's executable file. ➤ Make sure that your application is currently running and then click the pointing hand and point to the application on your desktop. **Note:** Sometimes a process is used only as a launcher that creates another process that actually provides the application functionality. Make sure that the executable file you provide is the one that actually provides the application functionality.
Remove button	Click the **Remove** button to remove the system counter definition from your test.

Option	Description
System Counter column	Defines the system counter you want to track for the selected application. Click inside the cell and then click the down arrow. Select the counter from the list. Click the expand button ⌄⌄ when displayed to show more counters. The system counters that can be monitored are the process counters accessible through the Performance Console (Select **Start** > **Run** > and then enter Perfmon). For information on the process counters accessible through the Performance Console, see the Performance Console Help.
Limit column	Defines the upper limit of the counter selected in the **System Counter** column. If the selected counter exceeds this value during the run session, the test fails. The limit value is optional. If you do not supply a value, the counter is tracked and the results are displayed in the Test Results window.
Description	Displays the description of the counter selected in the **System Counter** column, as provided by the Performance Console application.

46

Using the Setting Object to Set Testing Options During the Run Session

You can use the **Setting** object to control how QuickTest run tests by setting and retrieving testing options during a run session.

This chapter includes:

About Setting Testing Options During the Run Session

QuickTest testing options affect how you work with tests. For example, you can set the maximum time that QuickTest allows when loading a Web page, before determining that the URL address cannot be found.

You can set and retrieve the values of testing options during a run session using the Setting object in the Expert View. For more information on working in the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows."

By retrieving and setting testing options using the Setting object, you can control how QuickTest runs a test.

You can also set many testing options using the Options dialog box (global testing options) and the Test Settings dialog box (test-specific settings). For more information, see Chapter 44, "Setting Global Testing Options" and Chapter 45, "Setting Options for Individual Tests."

This chapter describes some of the QuickTest testing options that can be used with the Setting object from within a test script. For detailed information on all the available methods and properties for the Setting object, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

Note: You can also control QuickTest options as well as most other QuickTest operations using automation scripts. For more information, see "Automating QuickTest Operations" on page 1403 or the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

Setting Testing Options

You can use the Setting object to set the value of a testing option from within the test script. To set the option, use the following syntax:

Setting (*testing_option***) =** *new_value*

Some options are global and others affect only the current test. After you use a Setting object to set a testing option, the setting remains in effect until it is changed again or until the end of your current QuickTest session. You can also use the Setting object to change a setting for a specific part of a specific test. For more information see "Controlling the Test Run" on page 1305.

Some of the testing options that you can set using the Setting object are also available in the Options dialog box (global options) or the Test Settings dialog box (test specific settings). When you use the Setting object to set these options, the change is reflected in the relevant dialog box. Other test settings can be accessed using only one method, either the relevant dialog box or the Setting object.

Example: Using the Setting Object to Set an Option Reflected in the Options Dialog Box

If you run the following statement with the Web Add-in loaded:

```
Setting("AutomaticLinkRun")=1
```

QuickTest disables automatically created checkpoints in the test. The setting remains in effect during your current QuickTest session until it is changed again, either with another Setting statement, or by clearing the **Ignore automatic checkpoints while running tests or components** check box in the Web Advanced pane (Select **Tools** > **Options** > **Web** > **Advanced** node).

Example: Using the Setting Object to Set an Option Reflected in the Test Settings Dialog Box

If you run the following statement:

```
Setting("WebTimeOut")=50000
```

QuickTest automatically changes the amount of time it waits for a Web page to load before running a test step to 50000 milliseconds. The setting remains in effect during your current QuickTest session until it is changed again, either with another Setting statement, or by setting the **Browser Navigation Timeout** option in the Web pane of the Test Settings dialog box.

Note: Although the changes you make using the Setting object are reflected in the Options and Test Settings dialog boxes, these changes are not saved when you close QuickTest, unless you make other changes in the same dialog box manually and click **Apply** or **OK** (which saves all current settings in that dialog box).

Retrieving Testing Options

You can also use the Setting object to retrieve the current value of a testing option.

To store the value in a variable, use the syntax:

new_var = **Setting (** *testing_option* **)**

To display the value in a message box, use the syntax:

MsgBox (Setting (*testing_option***))**

For example:

```
LinkCheckSet = Setting("AutomaticLinkRun")
```

assigns the current value of the AutomaticLinkRun setting to the user-defined variable LinkCheckSet.

Other examples of testing options that you can use to retrieve a setting are shown in “Setting Testing Options” on page 1302.

Controlling the Test Run

You can use the retrieve and set capabilities of the Setting object together to control a run session without changing global settings. For example, if you want to change the DefaultTimeOut testing option to 5 seconds for objects on one Web page only, insert the following statement after the Web page opens in your test script:

```
'Keep the original value of the DefaultTimeOut testing option
old_delay = Setting ("DefaultTimeOut")

'Set temporary value for the DefaultTimeOut testing option
Setting("DefaultTimeOut")= 5000
```

To return the DefaultTimeOut testing option to its original value at the end of the Web page, insert the following statement immediately before linking to the next page in the script:

```
'Change the DefaultTimeOut testing option back to its original value.
Setting("DefaultTimeOut")=old_delay
```

Adding and Removing Run-Time Settings

In addition to the global and specific settings, you can also add, modify, and remove custom run-time settings. These settings are applicable during the run session only.

To add a new run-time setting, use the syntax:

Setting.Add "*testing_option*"**,** "*value*"

For example, you could create a setting that indicates the name of the current tester and displays the name in a message box.

```
Setting.Add "Tester Name", "Mark Train"
MsgBox Setting("Tester Name")
```

Tip: When using a Setting.Add statement, an error occurs if you try to add an existing setting option. To avoid this error you should use a Setting.Exists statement first. For more details about all the Setting methods, see the *HP QuickTest Professional Object Model Reference*.

To modify a run-time setting that has already been initialized, use the same syntax you use for setting any standard setting option:

Setting (*testing_option* **)** = *new_value*

For example:

```
Setting("Tester Name")="Alice Wonderlin"
```

To delete a custom run-time setting, use the following syntax:

Setting.Remove (*testing_option* **)**

For example:

```
Setting.Remove ("Tester Name")
```

Tip: When using a Setting.Remove statement, an error occurs if you try to remove a setting option that does not exist. To avoid this error you should use a Setting.Exists statement first. For more details about all the Setting methods, see the *HP QuickTest Professional Object Model Reference*.

Part X

Working with Advanced Testing Features

47

Learning Virtual Objects

You can teach QuickTest to recognize any area of your application as an object by defining it as a **virtual object**. Virtual objects enable you to create and run tests on objects that are not normally recognized by QuickTest.

You can manage the virtual objects defined on your computer using the Virtual Object Manager.

This chapter includes:

About Learning Virtual Objects

Your application may contain objects that behave like standard objects but are not recognized by QuickTest. You can define these objects as virtual objects and map them to standard classes, such as a button or a check box. QuickTest emulates the user's action on the virtual object during the run session. In the test results, the virtual object is displayed as though it is a standard class object.

For example, suppose you want to test a Web page containing a bitmap that the user clicks. The bitmap contains several different hyperlink areas, and each area opens a different destination page. When you create the test, the Web site matches the coordinates of the click on the bitmap and opens the destination page.

To enable QuickTest to click at the required coordinates during a run session, you can define a virtual object for an area of the bitmap, which includes those coordinates, and map it to the button class. When you run the test, QuickTest clicks the bitmap in the area defined as a virtual object so that the Web site opens the correct destination page.

You define a virtual object using the Virtual Object Wizard (**Tools > Virtual Objects > New Virtual Object**). The wizard prompts you to select the standard object class to which you want to map the virtual object. You then mark the boundaries of the virtual object using a crosshairs pointer. Next, you select a test object as the parent of the virtual object. Finally, you specify a name and a **collection** for the virtual object. For more information, see "Defining a Virtual Object" on page 1314.

Virtual object collections are groups of virtual objects that are stored in the Virtual Object Manager under a descriptive name. For more information, see "Understanding the Virtual Object Manager" on page 1312.

The virtual object collections displayed in the Virtual Object Manager are stored on your computer and not with the tests that contain virtual object steps. This means that if you use a virtual object in a test step, the object will be recognized during the run session only if it is run on a computer containing the appropriate virtual object definition. To copy your virtual object collection definitions to another computer, copy the contents of your <QuickTest installation folder>\dat\VoTemplate folder (or individual .vot collection files within this folder) to the same folder on the destination computer.

Note: QuickTest does not support virtual objects for analog or low-level recording. For more information on low-level recording, see "Creating Tests" on page 1552.

Understanding Virtual Objects

QuickTest identifies a virtual object according to its boundaries. Marking an object's boundaries specifies its size and position on a Web page or application window. When you assign a test object as the parent of your virtual object, you specify that the coordinates of the virtual object boundaries are relative to that parent object. When you record a test, QuickTest recognizes the virtual object within the parent object and adds it as a test object in the object repository so that QuickTest can identify the object during the run session. QuickTest also recognizes the virtual object as a test object when you add it manually to the object repository.

You can disable recognition of virtual objects without deleting them from the Virtual Object Manager. For more information, see "Removing or Disabling Virtual Object Definitions" on page 1327.

Notes:

- During a run session, make sure that the application window is the same size and in the same location as it was during recording, otherwise the coordinates of the virtual object relative to its parent object may be different, and this may affect the success of the run session.
- You can use virtual objects only when recording and running a test. You cannot insert any type of checkpoint on a virtual object, or use the Object Spy to view its properties.
- To perform an operation in the Active Screen on a marked virtual object, you must first record it, so that its properties are saved in the test object description in the object repository. If you perform an operation in the Active Screen on a virtual object that has not yet been recorded, QuickTest treats it as a standard object.

Understanding the Virtual Object Manager

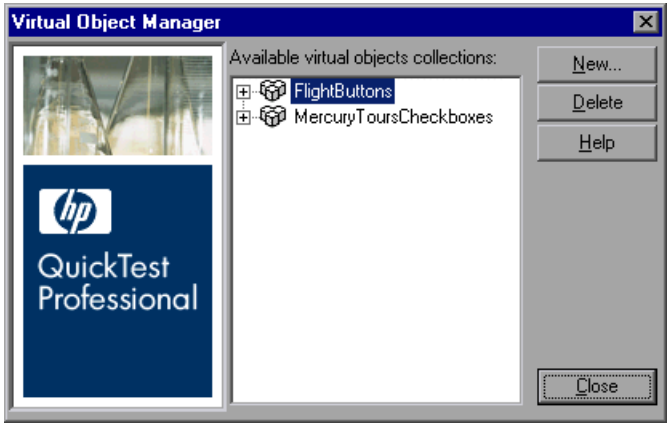
The Virtual Object Manager enables you to view and manage the virtual object collections defined on your computer. From the Virtual Object Manager, you can define and delete virtual objects and collections.

For more information on using the Virtual Object Manager, see The Virtual Object Manager Dialog Box.

The Virtual Object Manager Dialog Box

Description	Enables you to define and delete virtual objects and collections.
How to Access	Select **Tools** > **Virtual Objects** > **Virtual Object Manager.**
Learn More	**Conceptual overview**: "About Learning Virtual Objects" on page 1310 **Additional related topics**: "Understanding Virtual Objects" on page 1311

Below is an image of the Virtual Object Manager dialog box:



Virtual Object Manager Dialog Box Options

Option	Description
Available virtual object collections list	Displays the virtual object collections defined on your computer and the virtual objects contained in each collection. Click the + and - signs next to a collection to view or hide the virtual objects defined in that collection.
New button	Opens the Virtual Object Wizard, which guides you through the process of defining a new virtual object for a new or existing collection. For more information, see "Defining a Virtual Object" on page 1314.
Delete button	Deletes the selected virtual object or virtual object collection. For more information, see "Removing or Disabling Virtual Object Definitions" on page 1327.

Defining a Virtual Object

Using the Virtual Object Wizard, you can map a virtual object to a standard object class, specify the boundaries and the parent of the virtual object, and assign it a name. You can also group your virtual objects logically by assigning them to collections.

Note: You can define virtual objects only for objects on which QuickTest records Click or DblClick methods. Otherwise, the virtual object is ignored. For example, if you define a virtual object over the WinList object, the Select operation is recorded, and the virtual object is ignored. QuickTest does not support virtual objects for analog or low-level recording. For more information on low-level recording, see "Frequently Asked Questions" on page 1551.

For information on the Virtual Object Wizard screens, see:

- "The Virtual Object Wizard: Welcome Screen" on page 1317
- "The Virtual Object Wizard: Map to a Standard Class Screen" on page 1319
- "The Virtual Object Wizard: Mark Virtual Object Screen" on page 1321
- "The Virtual Object Wizard: Object Configuration Screen" on page 1323
- "The Virtual Object Wizard: Save Virtual Object Screen" on page 1325

To define a virtual object:

1. With QuickTest open (but not in record mode), open your application and display the object containing the area you want to define as a virtual object.
2. In QuickTest:
   a. Open the Virtual Object Wizard in one of the following ways:
      - Select **Tools** > **Virtual Objects** > **New Virtual Object**.
      - From the Virtual Object Manager, click **New**.
   b. Click **Next**.
3. In the Map to a Standard Class screen:
   a. Select a standard class to which you want to map your virtual object. For the **list** class, specify the number of rows in the virtual object. For the **table** class, select the number of rows and columns.

      For more information, see "The Virtual Object Wizard: Map to a Standard Class Screen" on page 1319.

   b. Click **Next**.
4. In the Mark Virtual Object screen:
   a. Click **Mark Object**. The QuickTest window and the Virtual Object Wizard are minimized.
   b. Use the crosshairs pointer to mark the area of the virtual object. You can use the arrow keys while holding down the left mouse button to make precise adjustments to the area you define with the crosshairs.

For more information, see "The Virtual Object Wizard: Mark Virtual Object Screen" on page 1321.

c Click **Next**.

Note: The virtual object should not overlap other virtual objects in your application. If the virtual object overlaps another virtual object, QuickTest may not record or run tests correctly on the virtual objects.

---

5 In the Object Configuration screen:

a Select an object in the object tree to assign it as the parent of the virtual object.

b In the **Identify object using** box, select how you want QuickTest to identify and map the virtual object.

For more information, see "The Virtual Object Wizard: Object Configuration Screen" on page 1323.

c Click **Next**.

6 In the Save Virtual Object screen:

a Specify a name and a collection for the virtual object. For more information, see "The Virtual Object Wizard: Save Virtual Object Screen" on page 1325.

b Perform one of the following:

- To add the virtual object to the Virtual Object Manager and close the wizard, select **No** and then click **Finish**.
- To add the virtual object to the Virtual Object Manager and define another virtual object, select **Yes** and then click **Next**. The wizard returns to the Map to a Standard Class screen, where you can define the next virtual object.

The Virtual Object Wizard: Welcome Screen

Description	Describes how you can use the wizard to define a virtual object by: ➤ Mapping it to a standard class ➤ Marking its boundaries ➤ Assigning a parent object ➤ Specifying a name and collection
How to Access	➤ Select **Tools** > **Virtual Objects** > **New Virtual Object**. ➤ Select **Tools** > **Virtual Objects** > **Virtual Object Manager.** From the Virtual Object Manager, click **New**.
Previous Screen	This is the first screen in the wizard.
Next Screen	"The Virtual Object Wizard: Map to a Standard Class Screen" on page 1319

Below is an image of the Virtual Object Wizard Welcome Screen:

Welcome Screen Options

Option	Description
Next	Click **Next** to go to the Map to a Standard Class screen.

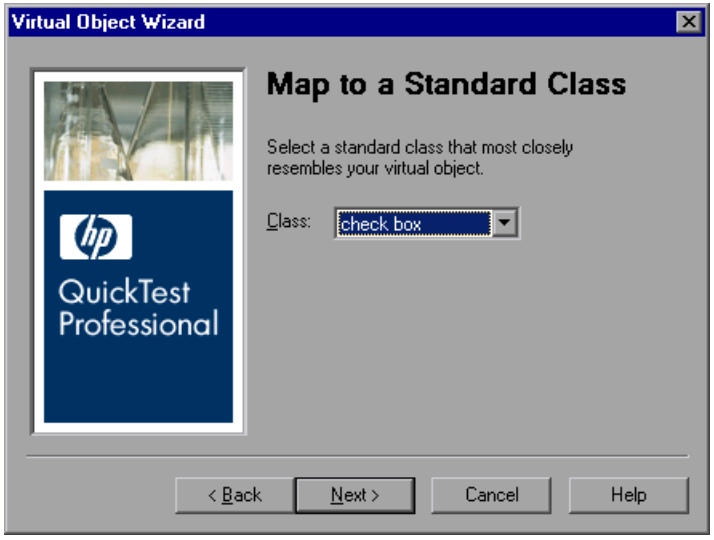
Additional References

Related Tasks	➤ "Defining a Virtual Object" on page 1314 ➤ "Removing or Disabling Virtual Object Definitions" on page 1327
Related Concepts	➤ "Understanding Virtual Objects" on page 1311 ➤ "The Virtual Object Manager Dialog Box" on page 1313

The Virtual Object Wizard: Map to a Standard Class Screen

Description	Enables you to configure a standard class for the virtual object.
Previous Screen	"The Virtual Object Wizard: Welcome Screen" on page 1317
Next Screen	"The Virtual Object Wizard: Save Virtual Object Screen" on page 1325

Below is an image of the Map to a Standard Class Screen:



Map to a Standard Class Screen Options

Option	Description
Class	Specify a standard object class from the list. ➤ For the list class, specify the number of rows in the virtual object. ➤ For the table class, select the number of rows and columns.
Back	Click **Back** to go to the Welcome screen.
Next	Click **Next** to go to the Mark Virtual Object screen.

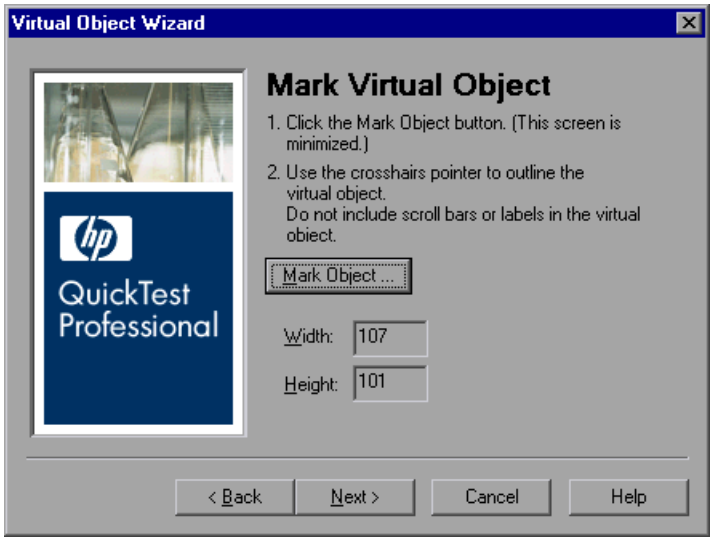
Additional References

Related Tasks	➤ "Defining a Virtual Object" on page 1314 ➤ "Removing or Disabling Virtual Object Definitions" on page 1327
Related Concepts	➤ "Understanding Virtual Objects" on page 1311 ➤ "The Virtual Object Manager Dialog Box" on page 1313

The Virtual Object Wizard: Mark Virtual Object Screen

Description	Enables you to configure the size and location of the virtual object.
Previous Screen	"The Virtual Object Wizard: Map to a Standard Class Screen" on page 1319
Next Screen	"The Virtual Object Wizard: Object Configuration Screen" on page 1323

Below is an image of the Mark Virtual Object Screen:



Mark Virtual Object Screen Options

Option	Description
Mark Object	Enables you to mark the outline for the virtual object. Using the crosshairs pointer, mark the outline for the virtual object in the application. Make sure that the virtual object does not overlap any other virtual object, as this may prevent QuickTest from identifying the virtual object during a run session. **Note:** To record and run tests properly, you must ensure that the application window is the same size and in the same position as it was when you defined the virtual object.
Width	Indicates the width of the outline in pixels.
Height	Indicates the height of the outline in pixels.
Back	Click **Back** to go to the Map to a Standard Class screen.
Next	Click **Next** to go to the Object Configuration screen.

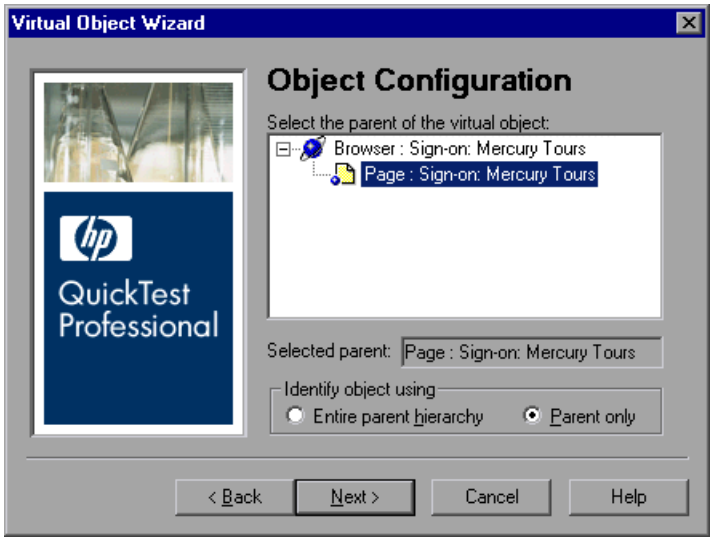
Additional References

Related Tasks	➤ "Defining a Virtual Object" on page 1314 ➤ "Removing or Disabling Virtual Object Definitions" on page 1327
Related Concepts	➤ "Understanding Virtual Objects" on page 1311 ➤ "The Virtual Object Manager Dialog Box" on page 1313

The Virtual Object Wizard: Object Configuration Screen

Description	Enables you to configure an object as a parent of the virtual object.
Previous Screen	"The Virtual Object Wizard: Mark Virtual Object Screen" on page 1321
Next Screen	"The Virtual Object Wizard: Save Virtual Object Screen" on page 1325

Below is an image of the Object Configuration Screen:



Object Configuration Screen Options

Option	Description
Select the parent of the virtual object area	Enables you to select an object in the tree as the parent object. The coordinates of the virtual object outline are relative to the parent object.
Selected parent box (read-only)	Indicates the name of the object selected as the parent object.

Option	Description
Identify object using area	Indicates how you want QuickTest to identify and map the virtual object, as described below. The coordinates of the virtual object outline are relative to the parent object you select.
Entire parent hierarchy radio button	Select this option to identify the virtual object in one occurrence only. QuickTest identifies the virtual object only if it has the exact parent hierarchy. For example, if the virtual object is defined using Browser("A").Page("B").Image("C"), QuickTest does not recognize it if the hierarchy changes to Browser("X").Page("B").Image("C").
Parent only radio button	Select this option to identify all occurrences of the virtual object. QuickTest identifies the virtual object using its direct parent only, regardless of the entire parent hierarchy. For example, if the virtual object was defined using Browser("A").Page("B").Image("C"), QuickTest will recognize the virtual object even if the hierarchy changes to Browser("X").Page("Y").Image("C").
Back button	Click **Back** to go to the Mark Virtual Object screen.
Next button	Click **Next** to go to the Save Virtual Object screen.

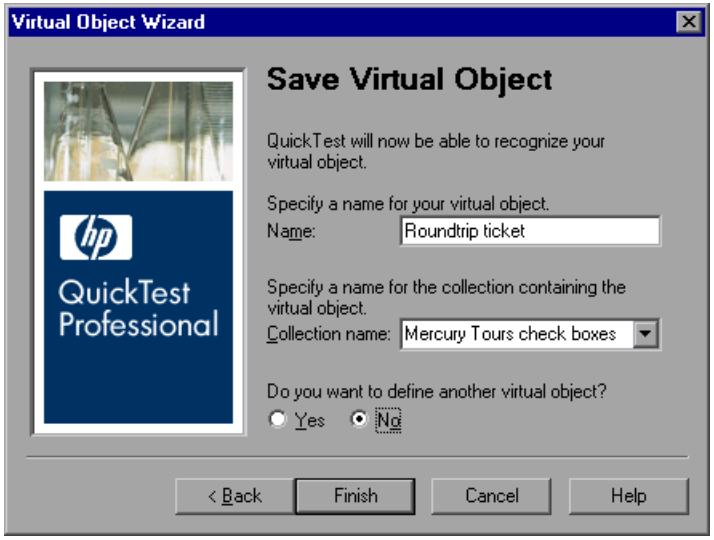
Additional References

Related Tasks	➤ "Defining a Virtual Object" on page 1314 ➤ "Removing or Disabling Virtual Object Definitions" on page 1327
Related Concepts	➤ "Understanding Virtual Objects" on page 1311 ➤ "The Virtual Object Manager Dialog Box" on page 1313

The Virtual Object Wizard: Save Virtual Object Screen

Description	Enables you to configure a name and a collection for the virtual object. Also enables you to begin defining another virtual object.
Previous Screen	"The Virtual Object Wizard: Object Configuration Screen" on page 1323
Next Screen	This is the final screen in the wizard.

Below is an image of the Save Virtual Object Screen:



Save Virtual Object Screen Options

Option	Description
Name box	Specify the name of the virtual object. Choose from the list of collections or create a new one by entering a new name in the **Collection name** box.
Collection name box	Specify the name of the collection. You can choose an existing name or enter a new name.

Option	Description
Do you want to define another virtual object? area	➤ Select **Yes** and then click **Next** to use the wizard to define another virtual object. ➤ Select **No** and then click **Finish** to close the wizard.
Back button	Click **Back** to go to the Object Configuration screen.
Next button	If you selected the **Yes** radio button for the **Do you want to define another virtual object?** option, QuickTest displays the **Next** button. Click **Next** to save the virtual object and go to the Map to a Standard Class screen to begin defining another virtual object.
Finish button	If you selected the **No** radio button for the **Do you want to define another virtual object?** option, QuickTest displays the **Finish** button. Click **Finish** to save the virtual object and close the wizard.

Additional References

Related Tasks	➤ "Defining a Virtual Object" on page 1314 ➤ "Removing or Disabling Virtual Object Definitions" on page 1327
Related Concepts	➤ "Understanding Virtual Objects" on page 1311 ➤ "The Virtual Object Manager Dialog Box" on page 1313

Removing or Disabling Virtual Object Definitions

You can remove virtual objects from your test by deleting them or by disabling recognition of these objects while recording.

To delete a virtual object:

1 Select **Tools** > **Virtual Objects** > **Virtual Object Manager.** The Virtual Object Manager opens. For more information, see "The Virtual Object Manager Dialog Box" on page 1313.

2 In the list of available virtual object collections, click the plus sign (+) next to the collection to display the virtual object you want to delete. Select the virtual object, and click **Delete**.

To delete an entire collection, select it and click **Delete**.

3 Click **Close**.

Tip: Click **New** in the Virtual Object Manager to open the Virtual Object Wizard, where you can define a new virtual object.

To disable recognition of virtual objects while recording:

1 Select **Tools** > **Options** or click the **Options** toolbar button. The Options dialog box opens.

2 In the General pane, select the **Disable recognition of virtual objects while recording** check box.

3 Click **OK**.

Note: When you want QuickTest to recognize virtual objects during recording, ensure that the **Disable recognition of virtual objects while recording** check box in the General pane of the Options dialog box is cleared. For more information, see "Setting General Testing Options" on page 1234.

48

Defining and Using Recovery Scenarios

You can instruct QuickTest to recover from unexpected events and errors that occur in your testing environment during a run session.

This chapter includes:

About Defining and Using Recovery Scenarios

Unexpected events, errors, and application crashes during a run session can disrupt your run session and distort results. This is a problem particularly when tests run unattended—the test pauses until you perform the operation needed to recover. To handle situations such as these, QuickTest enables you to create recovery scenarios and associate them with specific tests. Recovery scenarios activate specific recovery operations when trigger events occur. For information on when to use recovery scenarios, see "Deciding When to Use Recovery Scenarios" on page 1332.

The Recovery Scenario Manager provides a wizard that guides you through the process of defining a recovery scenario, which includes a definition of an unexpected event and the operations necessary to recover the run session. For example, you can instruct QuickTest to detect a **Printer out of paper** message and recover the run session by clicking the **OK** button to close the message and continue the test.

A recovery scenario consists of the following:

- **Trigger Event.** The event that interrupts your run session. For example, a window that may pop up on screen, or a QuickTest run error.
- **Recovery Operations.** The operations to perform to enable QuickTest to continue running the test after the trigger event interrupts the run session. For example, clicking an **OK** button in a pop-up window, or restarting Microsoft Windows.
- **Post-Recovery Test Run Option.** The instructions on how QuickTest should proceed after the recovery operations have been performed, and from which point in the test QuickTest should continue, if at all. For example, you may want to restart a test from the beginning, or skip a step entirely and continue with the next step in the test.

Recovery scenarios are saved in recovery scenario files. A recovery scenario file is a logical collection of recovery scenarios, grouped according to your own specific requirements.

To instruct QuickTest to perform a recovery scenario during a run session, you must first associate the recovery scenario with that test. A test can have any number of recovery scenarios associated with it. You can prioritize the scenarios associated with your test to ensure that trigger events are recognized and handled in the required order. For more information, see "Adding Recovery Scenarios to Your Test" on page 1373.

When you run a test for which you have defined recovery scenarios and an error occurs, QuickTest looks for the defined trigger events that caused the error. If a trigger event has occurred, QuickTest performs the corresponding recovery and post-recovery operations.

You can also control and activate your recovery scenarios during the run session by inserting Recovery statements into your test. For more information, see "Programmatically Controlling the Recovery Mechanism" on page 1379.

Note: If you select **On error** in the **Activate recovery scenarios** box in the Recovery pane of the Test Settings dialog box, the recovery mechanism does not handle triggers that occur in the last step of a test. If you chose this option and need to recover from an unexpected event or error that may occur in the last step of a test, you can do this by adding an extra step to the end of your test.

Deciding When to Use Recovery Scenarios

Recovery scenarios are intended for use **only** with events that you cannot predict in advance, or for events that you cannot otherwise synchronize with a specific step in your test. For example, you could define a recovery scenario to handle printer errors. Then if a printer error occurs during a run session, the recovery scenario could instruct QuickTest to click the default button in the Printer Error message box.

You would use a recovery scenario in this example because you cannot handle this type of error directly in your test. This is because you cannot know at what point the network will return the printer error. Even if you try to handle this event by adding an If statement in your test immediately after a step that sends a file to the printer, your test may progress several steps before the network returns the actual printer error.

If you can predict that a certain event may happen at a specific point in your test, it is highly recommended to handle that event directly within your test by adding steps such as If statements or optional steps, rather than depending on a recovery scenario. For example, if you know that an Overwrite File message box may open when a **Save** button is clicked during a run session, you can handle this event with an If statement that clicks **OK** if the message box opens or by adding an optional step that clicks **OK** in the message box.

Handling an event directly within your test enables you to handle errors more specifically than recovery scenarios, which by nature are designed to handle a more generic set of unpredictable events. It also enables you to control the timing of the corrective operation with minimal resource usage and maximum performance. By default, recovery scenario operations are activated only after a step returns an error. This can potentially occur several steps after the step that originally caused the error. The alternative, checking for trigger events after every step, may slow performance. For this reason, it is best to handle predictable errors directly in your test.

For more information on optional steps, see “Using Optional Steps” on page 963. For more information on inserting programming statements such as If statements, see Chapter 28, “Adding Steps Containing Programming Logic.”

Defining Recovery Scenarios

You create recovery scenarios using the Recovery Scenario Wizard (accessed from the Recovery Scenario Manager dialog box). The Recovery Scenario Wizard leads you through the process of defining each of the stages of a recovery scenario. As you create your recovery scenarios, you save them in a recovery file. A recovery file is a convenient way to organize and store multiple recovery scenarios together.

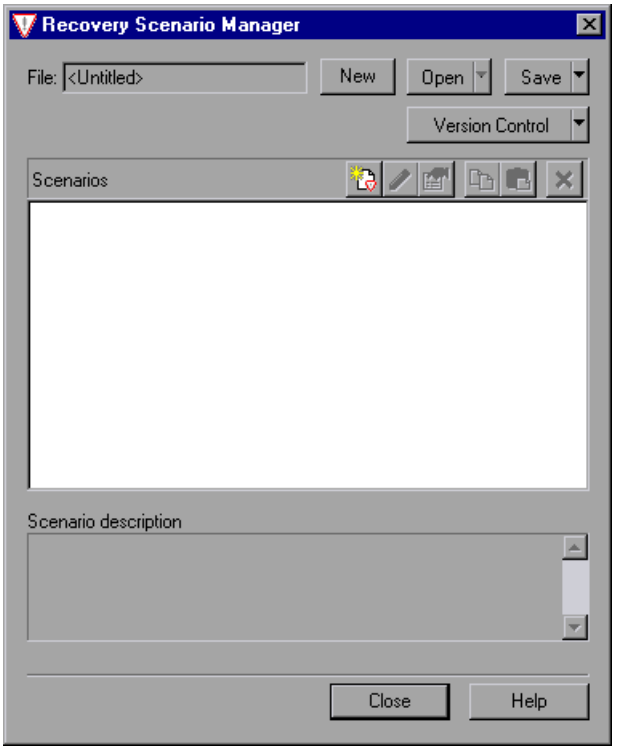
Using the Recovery Scenario Manager dialog box, you can then select any recovery file to manage all of the recovery scenarios stored in that file. This enables you to edit a selected recovery scenario, associate specific recovery scenarios with specific tests to instruct QuickTest to implement the recovery scenarios when specified trigger events occur, and so forth.

Creating a Recovery File

You create recovery files to store your recovery scenarios. You can create a new recovery file or edit an existing one.

To create a recovery file:

1 Select **Resources** > **Recovery Scenario Manager**. The Recovery Scenario Manager dialog box opens.



2 By default, the Recovery Scenario Manager dialog box opens with a new recovery file. You can use this new file, or you can open an existing recovery file, in one of the following ways:

➤ Click the arrow next to the **Open** button to select a recently-used recovery file from the list.

➤ Do the following:

a Click the **Open** button to choose an existing recovery file.

b In the sidebar of the Open Recovery Scenario dialog box, select the location where the file is stored, for example, File System or Quality Center Test Resources.

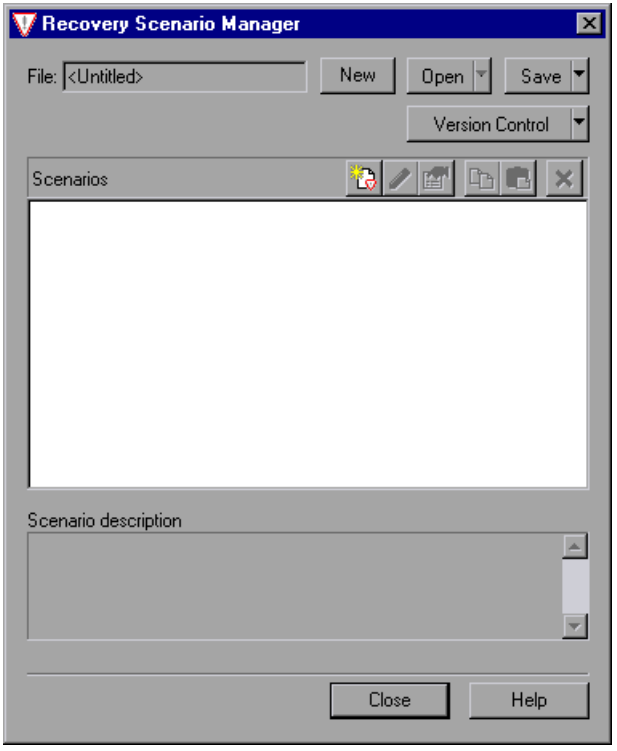
c Browse to and select the recovery scenario file you want to open and click **Open**. If the recovery file is stored in a version-control-enabled project in Quality Center, you can click the **Open** down arrow and select **Open and Check out** to check out the file.

You can now create recovery scenarios using the Recovery Scenario Wizard and save them in your recovery file, as described in the following sections.

Understanding the Recovery Scenario Manager Dialog Box

The Recovery Scenario Manager dialog box enables you to create and edit recovery files, and create and manage the recovery scenarios stored in those files.

The Recovery Scenario Manager dialog box displays the name of the currently open recovery file, a list of the scenarios saved in the recovery file, and a description of each scenario.



The Recovery Scenario Manager dialog box contains the following toolbar buttons:

Option	Description
	Creates a new recovery file. For more information, see "Creating a Recovery File" on page 1334.
	Opens an existing recovery file. You can also click the arrow to select a recovery file from the list of recently-used recovery files.
	Saves the current recovery file. For more information, see "Saving the Recovery Scenario in a Recovery File" on page 1365.
	Enables you to manage version control for your recovery scenarios. (Options are available only if QuickTest is connected to a version control-enabled Quality Center project.) For more information, see "Managing Versions of Assets in Quality Center" on page 1480 and "Viewing and Comparing Versions of QuickTest Assets" on page 1461.
	Opens the Recovery Scenario Wizard, in which you define a new recovery scenario. For more information, see "Understanding the Recovery Scenario Wizard" on page 1338.
	Opens the Recovery Scenario Wizard for the selected recovery scenario, in which you can modify the recovery scenario settings. For more information, see "Modifying Recovery Scenarios" on page 1370.
	Displays summary properties for the selected recovery scenario in read-only format. For more information, see "Viewing Recovery Scenario Properties" on page 1368.
	Copies a recovery scenario from the open recovery file to the Clipboard. This enables you to paste a recovery scenario into another recovery file. For more information, see "Copying Recovery Scenarios between Recovery Scenario Files" on page 1371.

Option	Description
	Pastes a recovery scenario from the Clipboard into the open recovery file. For more information, see “Copying Recovery Scenarios between Recovery Scenario Files” on page 1371.
	Deletes a recovery scenario. For more information, see “Deleting Recovery Scenarios” on page 1370.

Note: Each recovery scenario is represented by an icon that indicates its type. For more information, see “Managing Recovery Scenarios” on page 1367.

Understanding the Recovery Scenario Wizard

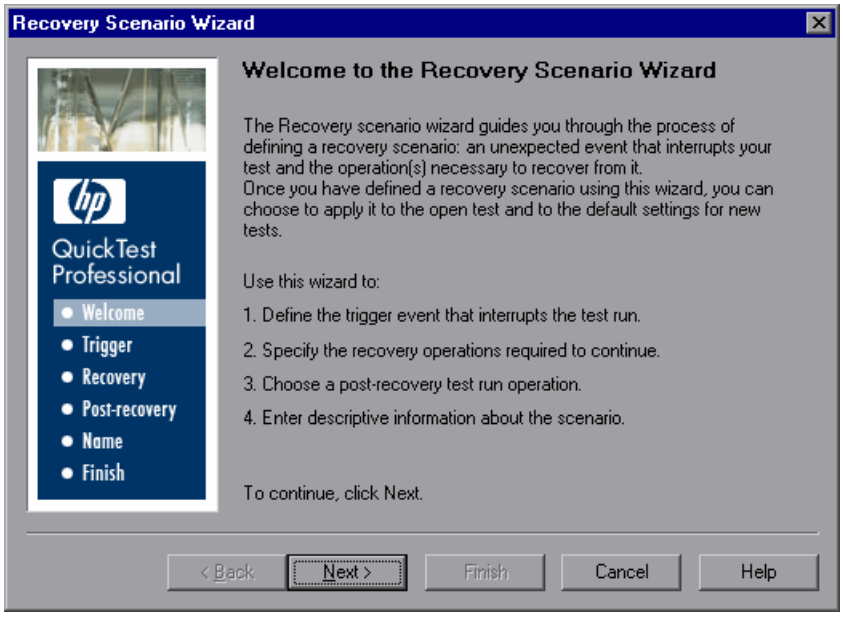
The Recovery Scenario Wizard leads you, step-by-step, through the process of creating a recovery scenario. The Recovery Scenario Wizard contains the following main steps:

- Defining the trigger event that interrupts the run session
- Specifying the recovery operations required to continue
- Choosing a post-recovery test run operation
- Specifying a name and description for the recovery scenario
- Specifying whether to associate the recovery scenario to the current test and/or to all new tests

You open the Recovery Scenario Wizard by clicking the **New Scenario** button in the Recovery Scenario Manager dialog box (**Resources** > **Recovery Scenario Manager**).

Welcome to the Recovery Scenario Wizard Screen

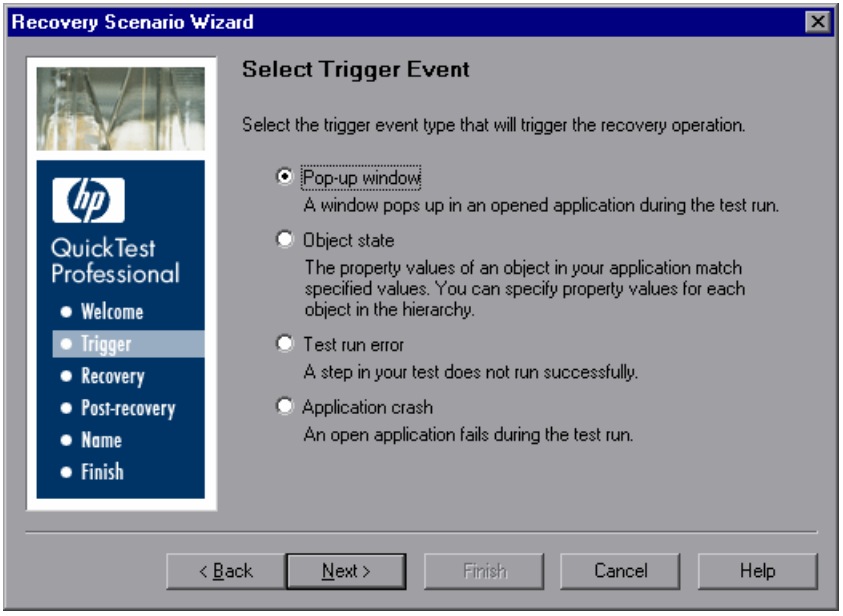
The Welcome to the Recovery Scenario Wizard screen provides general information on the different options in the Recovery Scenario Wizard, and provides an overview of the stages involved in defining a recovery scenario.



Click **Next** to continue to the Select Trigger Event Screen (described on page 1340).

Select Trigger Event Screen

The Select Trigger Event screen enables you to define the event type that triggers the recovery scenario, and the way in which QuickTest recognizes the event.



Select a type of trigger and click **Next**. The next screen displayed in the wizard depends on which of the following trigger types you select:

➤ **Pop-up window.** QuickTest detects a pop-up window and identifies it according to the window title and textual content. For example, a message box may open during a run session, indicating that the printer is out of paper. QuickTest can detect this window and activate a defined recovery scenario to continue the run session.

Select this option and click **Next** to continue to the Specify Pop-up Window Conditions Screen (described on page 1342).

➤ **Object state.** QuickTest detects a specific test object state and identifies it according to its property values and the property values of all its ancestors. Note that an object is identified only by its property values, and not by its class.

For example, a specific button in a dialog box may be disabled when a specific process is open. QuickTest can detect the object property state of the button that occurs when this problematic process is open and activate a defined recovery scenario to close the process and continue the run session.

Select this option and click **Next** to continue to the Select Object Screen (described on page 1345).

➤ **Test run error.** QuickTest detects a run error and identifies it by a failed return value from a method. For example, QuickTest may not be able to identify a menu item specified in the method argument, due to the fact that the menu item is not available at a specific point during the run session. QuickTest can detect this run error and activate a defined recovery scenario to continue the run session.

Select this option and click **Next** to continue to the Select Test Run Error Screen (described on page 1349).

➤ **Application crash.** QuickTest detects an application crash and identifies it according to a predefined list of applications. For example, a secondary application may crash when a certain step is performed in the run session. You want to be sure that the run session does not fail because of this crash, which may indicate a different problem with your application. QuickTest can detect this application crash and activate a defined recovery scenario to continue the run session.

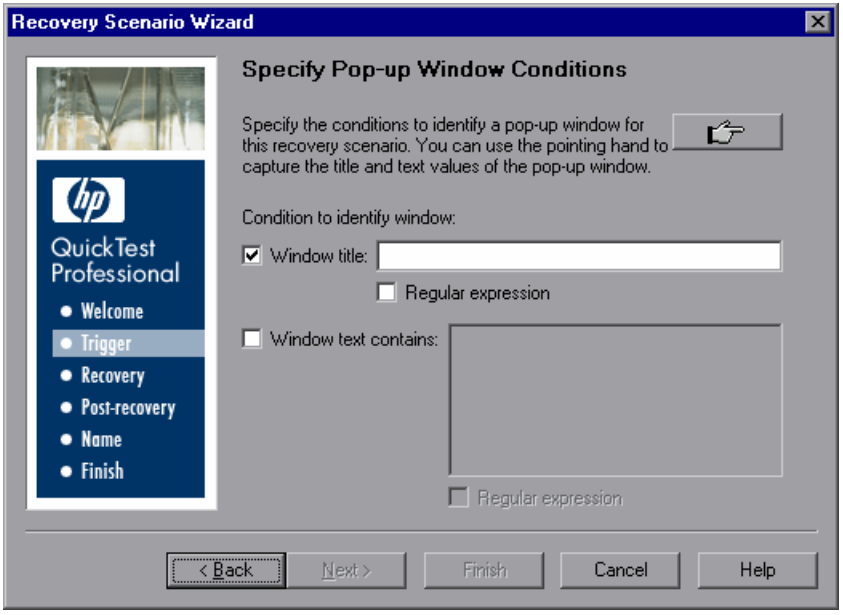
Select this option and click **Next** to continue to the Recovery Operations Screen (described on page 1352).

Notes:

- The set of recovery operations is performed for each occurrence of the trigger event criteria. For example, suppose you define a specific object state, and two objects match this state, the set of recovery operations is performed two times, once for each object that matches the specified state.
- The recovery mechanism does not handle triggers that occur in the last step of a test. If you need to recover from an unexpected event or error that may occur in the last step of a test, you can do this by adding an extra step to the end of your test.

Specify Pop-up Window Conditions Screen

If you chose a **Pop-up window** trigger in the Select Trigger Event Screen (described on page 1340), the Specify Pop-up Window Conditions screen opens.



Perform one of the following to specify how the pop-up window should be identified:

- Choose whether you want to identify the pop-up window according to its **Window title** and/or **Window text** and then enter the text used to identify the pop-up window. You can use regular expressions in the window title or textual content by selecting the relevant **Regular expression** check box and then entering the regular expression in the relevant location. For information on regular expressions, see "Understanding and Using Regular Expressions" on page 762.
- Click the pointing hand. Then click the pop-up window to capture the window title and textual content of the window. For more information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1344.

Note: Using the first option (**Window title** and/or **Window text**) instructs QuickTest to identify any pop-up window that contains the relevant title and/or text. Using the second option (pointing hand) instructs QuickTest to identify only pop-up windows that match the object property values of the window you select.

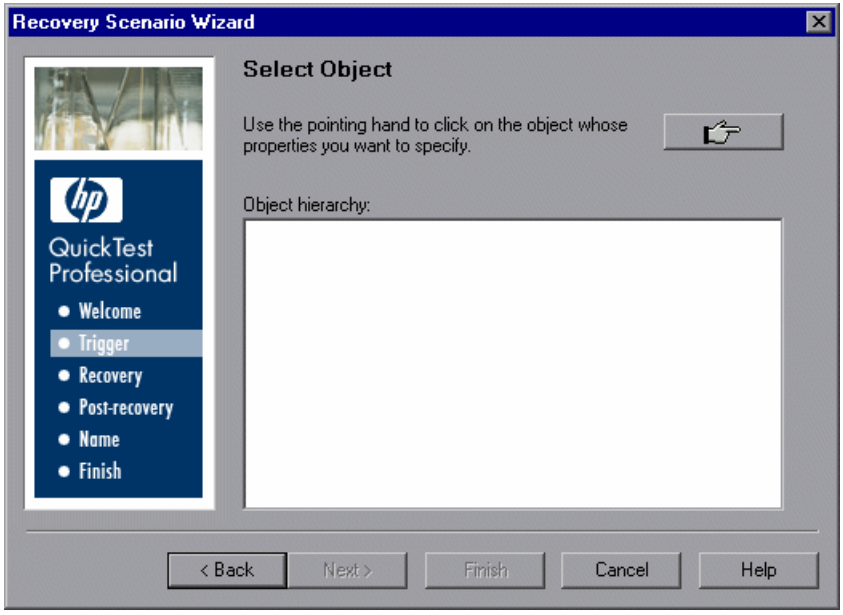
Click **Next** to continue to the Recovery Operations Screen (described on page 1352).

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Select Object Screen

If you chose an **Object state** trigger in the Select Trigger Event Screen (described on page 1340), the Select Object screen opens.



Click the pointing hand and then click the object whose properties you want to specify. For more information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1347.

If the location you click is associated with more than one object, the Object Selection–Object State Trigger dialog box opens.

Select the object whose properties you want to specify and click **OK**. The selected object and its parents are displayed in the Select Object screen.

Note: The hierarchical object selection tree also enables you to select an object that QuickTest would not ordinarily learn (a non-parent object), such as a Web table.

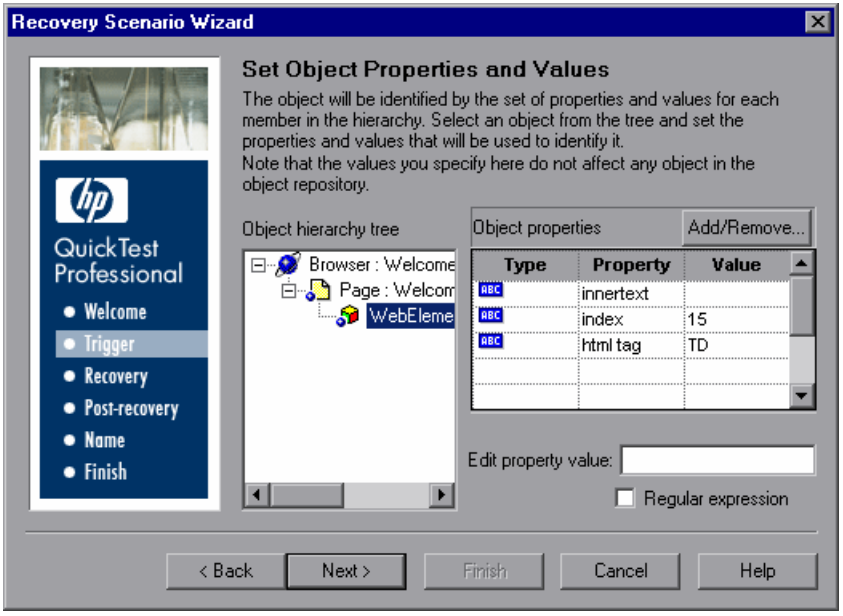
Click **Next** to continue to the Set Object Properties and Values Screen (described on page 1348).

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Set Object Properties and Values Screen

After you select the object whose properties you want to specify in the Select Object Screen (described on page 1345), the Set Object Properties and Values screen opens.



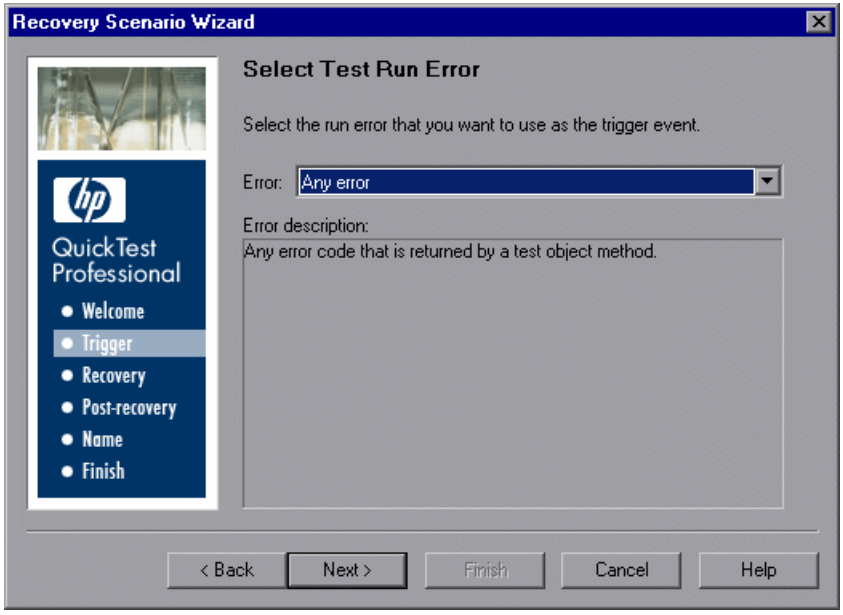
For each object in the hierarchy, in the **Edit property value** box, you can modify the property values used to identify the object. You can also click the **Add/Remove** button to add or remove object properties from the list of property values to check. Note that an object is identified only by its property values, and not by its class.

Select the **Regular expression** check box if you want to use regular expressions in the property value. For information on regular expressions, see “Understanding and Using Regular Expressions” on page 762.

Click **Next** to continue to the Recovery Operations Screen (described on page 1352).

Select Test Run Error Screen

If you chose a **Test run error** trigger in the Select Trigger Event Screen (described on page 1340), the Select Test Run Error screen opens.



In the **Error** list, select the run error that you want to use as the trigger event:

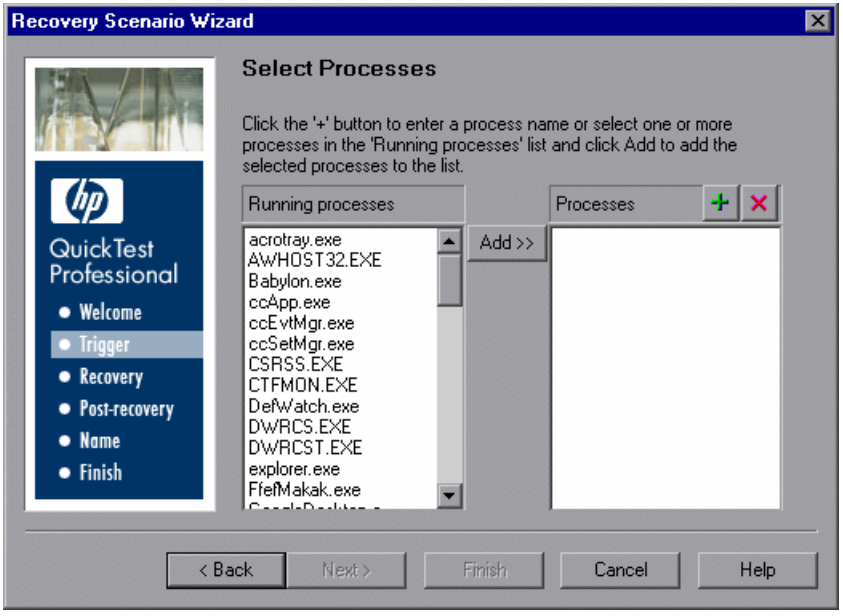
- **Any error.** Any error code that is returned by a test object method.
- **Item in list or menu is not unique.** Occurs when more than one item in the list, menu, or tree has the name specified in the method argument.
- **Item in list or menu not found.** Occurs when QuickTest cannot identify the list, menu, or tree item specified in the method argument. This may be due to the fact that the item is not currently available or that its name has changed.
- **More than one object responds to the physical description.** Occurs when more than one object in your application has the same property values as those specified in the test object description for the object specified in the step.

- **Object is disabled.** Occurs when QuickTest cannot perform the step because the object specified in the step is currently disabled.
- **Object not found.** Occurs when no object within the specified parent object matches the test object description for the object.
- **Object not visible.** Occurs when QuickTest cannot perform the step because the object specified in the step is not currently visible on the screen.

Click **Next** to continue to the "Recovery Operations Screen" on page 1352.

Select Processes Screen

If you chose an **Application crash** trigger in the Select Trigger Event Screen (described on page 1340), the Select Processes screen opens.



The **Running processes** list displays all application processes that are currently running. The **Processes** list displays the application processes that will trigger the recovery scenario if they crash.

You can add application processes to the **Processes** list by typing them in the **Processes** list or by selecting them from the **Running processes** list.

- To add a process from the **Running processes** list, double-click a process in the **Running processes** list or select it and click the **Add** button. You can select multiple processes using standard Windows multiple selection techniques (CTRL and SHIFT keys).

- To add a process directly to the **Processes** list, click the **Add New Process** button to enter the name of any process you want to add to the list.

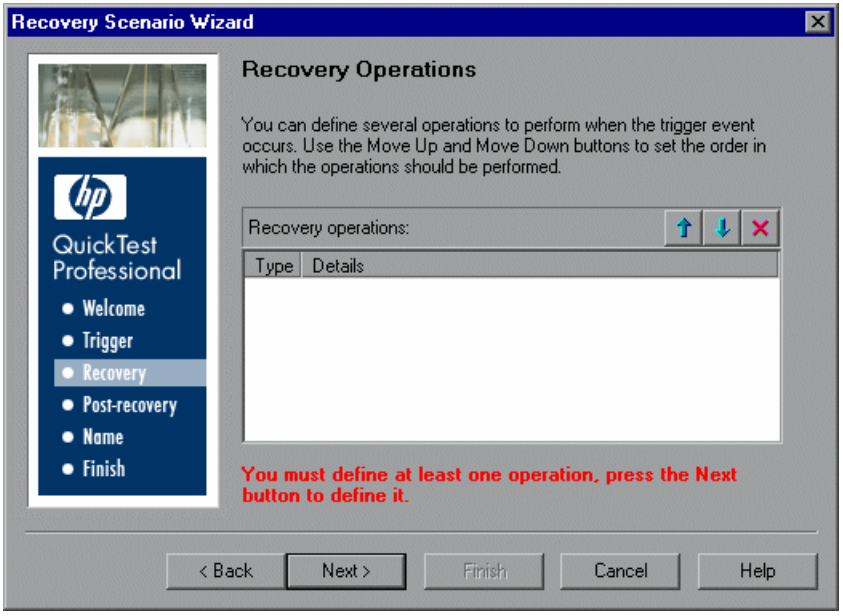
- To remove a process from the **Processes** list, select it and click the **Remove Process** button.

Tip: You can modify the name of a process by selecting it in the **Processes** list and clicking the process name to edit it.

Click **Next** to continue to the Recovery Operations Screen (described on page 1352).

Recovery Operations Screen

The Recovery Operations screen enables you to manage the collection of recovery operations in the recovery scenario. Recovery operations are operations that QuickTest performs sequentially when it recognizes the trigger event.



You must define at least one recovery operation. To define a recovery operation and add it to the **Recovery operations** list, click **Next** to continue to the Recovery Operation Screen (described on page 1353).

If you define two or more recovery operations, you can select a recovery operation and use the **Move Up** or **Move Down** buttons to change the order in which QuickTest performs the recovery operations. You can also select a recovery operation and click the **Remove** button to delete a recovery operation from the recovery scenario.

Note: If you define a **Restart Microsoft Windows** recovery operation, it is always inserted as the last recovery operation, and you cannot change its position in the list.

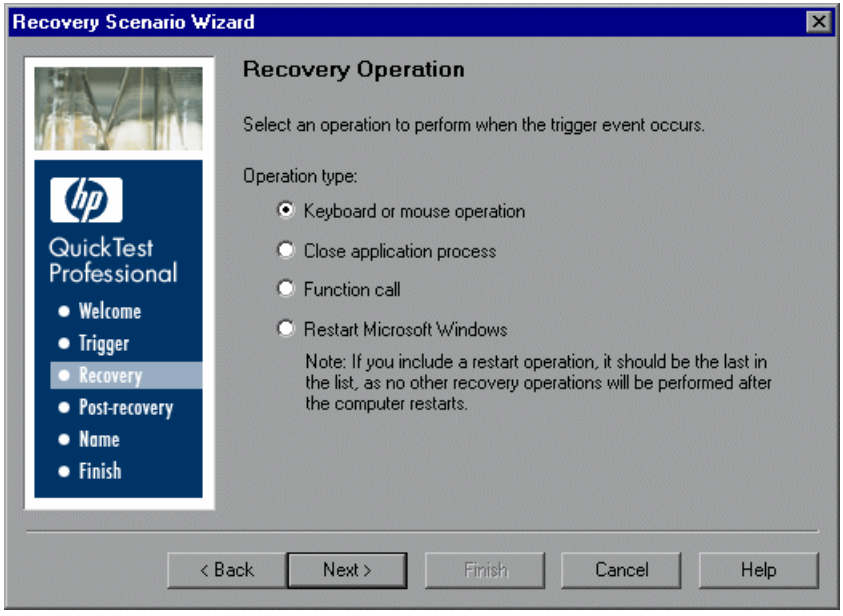
After you have defined at least one recovery operation, the **Add another recovery operation** check box is displayed.

➤ Select the check box and click **Next** to define another recovery operation.

➤ Clear the check box and click **Next** to continue to the Post-Recovery Test Run Options Screen (described on page 1361).

Recovery Operation Screen

The Recovery Operation screen enables you to specify the operations QuickTest performs after it detects the trigger event.



Select a type of recovery operation and click **Next**. The next screen displayed in the wizard depends on which recovery operation type you select.

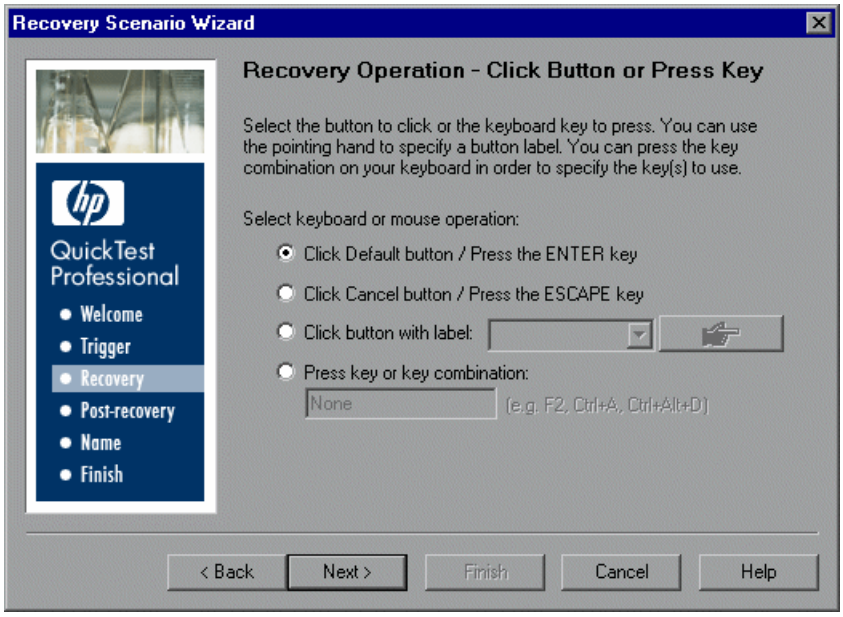
You can define the following types of recovery operations:

- **Keyboard or mouse operation.** QuickTest simulates a click on a button in a window or a press of a keyboard key. Select this option and click **Next** to continue to the Recovery Operation - Click Button or Press Key Screen (described on page 1355).
- **Close application process.** QuickTest closes specified processes. Select this option and click **Next** to continue to the Recovery Operation - Close Processes Screen (described on page 1357).
- **Function call.** QuickTest calls a VBScript function. Select this option and click **Next** to continue to the Recovery Operation - Function Call Screen (described on page 1358).
- **Restart Microsoft Windows.** QuickTest restarts Microsoft Windows. Select this option and click **Next** to continue to the Recovery Operations Screen (described on page 1352).

Note: If you use the **Restart Microsoft Windows** recovery operation, you must ensure that any test associated with this recovery scenario is saved before you run it. You must also configure the computer on which the test is run to automatically log in on restart.

Recovery Operation - Click Button or Press Key Screen

If you chose a **Keyboard or mouse operation** recovery operation in the Recovery Operation Screen (described on page 1353), the Recovery Operation – Click Button or Press Key screen opens.



Specify the keyboard or mouse operation that you want QuickTest to perform when it detects the trigger event:

- **Click Default button / Press the ENTER key.** Instructs QuickTest to click the default button or press the ENTER key in the displayed window when the trigger occurs.
- **Click Cancel button / Press the ESCAPE key.** Instructs QuickTest to click the **Cancel** button or press the ESCAPE key in the displayed window when the trigger occurs.

- **Click button with label.** Instructs QuickTest to click the button with the specified label in the displayed window when the trigger occurs. If you select this option, click the pointing hand and then click anywhere in the trigger window. For more information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1356.

  All button labels in the selected window are displayed in the list box. Select the required button from the list.

- **Press key or key combination.** Instructs QuickTest to press the specified keyboard key or key combination in the displayed window when the trigger occurs. If you select this option, click in the edit box and then press the key or key combination on your keyboard that you want to specify.

  Click **Next**. The Recovery Operations Screen reopens, showing the keyboard or mouse recovery operation that you defined.

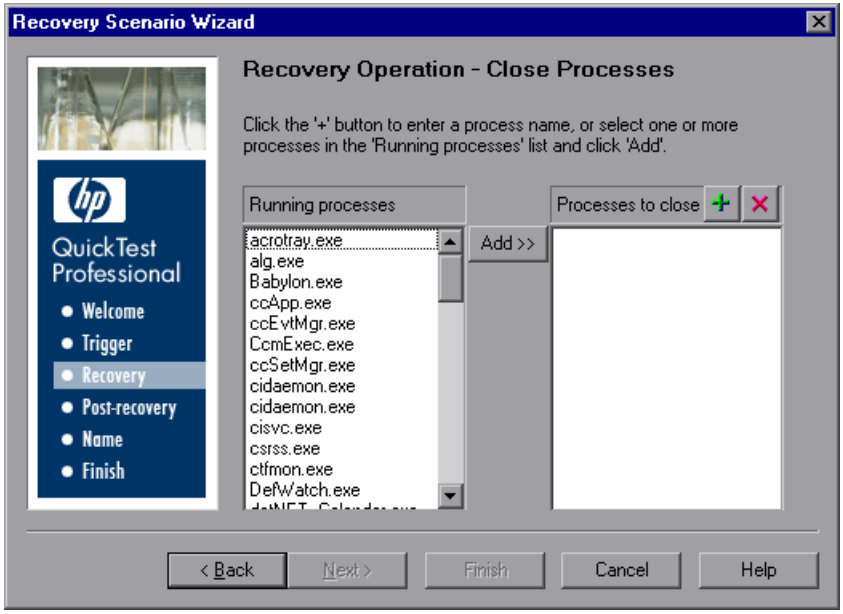
Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

➤ If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Recovery Operation - Close Processes Screen

If you chose a **Close application process** recovery operation in the Recovery Operation Screen (described on page 1353), the Recovery Operation – Close Processes screen opens.



The **Running processes** list displays all application processes that are currently running. The **Processes to close** list displays the application processes that will be closed when the trigger is activated.

➤ To add a process from the **Running processes** list, double-click a process in the **Running processes** list or select it and click the **Add** button. You can select multiple processes using standard Windows multiple selection techniques (CTRL and SHIFT keys).

➤ To add a process directly to the **Processes to close** list, click the **Add New Process** button to enter the name of any process you want to add to the list.

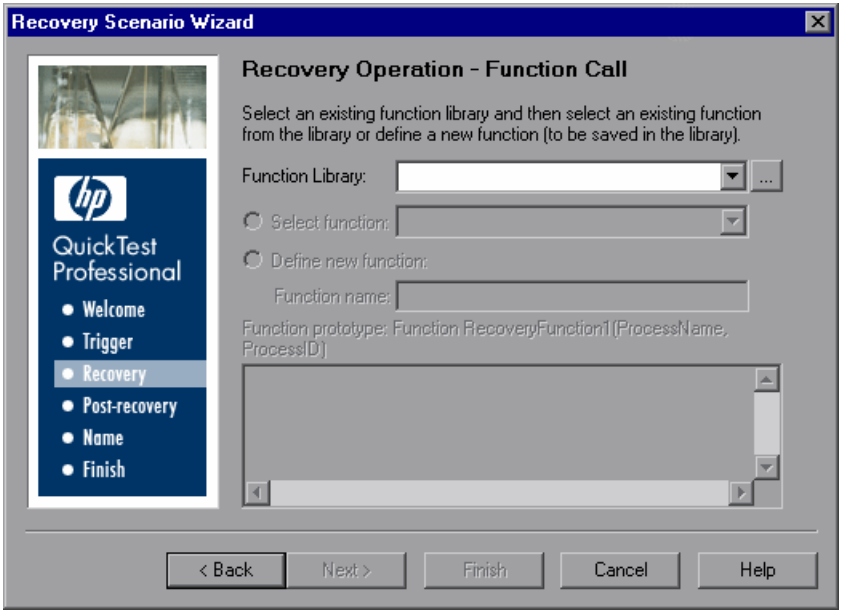
➤ To remove a process from the **Processes to close** list, select it and click the **Remove Process** button.

Tip: You can modify the name of a process by selecting it in the **Processes to close** list and clicking the process name to edit it.

Click **Next**. The Recovery Operations Screen reopens, showing the close processes recovery operation that you defined.

Recovery Operation - Function Call Screen

If you chose a **Function call** recovery operation in the Recovery Operation Screen (described on page 1353), the Recovery Operation – Function Call screen opens.



Select a recently specified function library in the **Function Library** box. Alternatively, click the browse button to navigate to an existing function library.

Note: QuickTest automatically associates the function library you select with your test. Therefore, you do not need to associate the function library with your test in the Resources pane of the Test Settings dialog box.

After you select a function library, choose one of the following options:

➤ **Select function.** Choose an existing function from the function library you selected.

Only functions that match the prototype syntax for the trigger type selected in the "Select Trigger Event Screen" on page 1340 are displayed.

Following is the prototype for each trigger type:

Test run error trigger

```
OnRunStep
(
[in]  Object as Object: The object of the current step.
[in]  Method as String: The method of the current step.
[in]   Arguments as Array: The actual method's arguments.
[in]   Result as Integer: The actual method's result.
)
```

Pop-up window and Object state triggers

```
OnObject
(
[in]  Object as Object: The detected object.
)
```

Application crash trigger

```
OnProcess
(
[in]  ProcessName as String: The detected process's Name.
[in]  ProcessId as Integer: The detected process' ID.
)
```

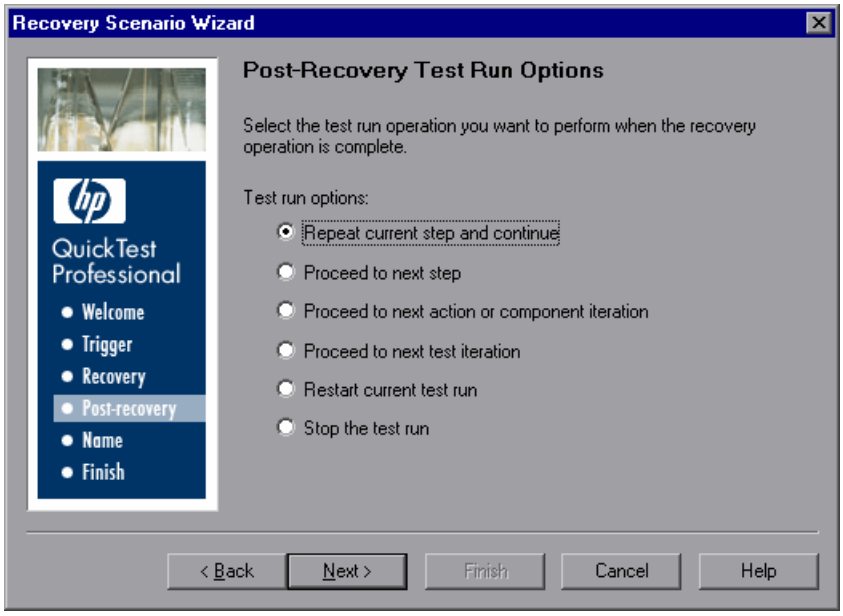
➤ **Define new function.** Create a new function by specifying a unique name for it, and defining the function in the **Function Name** box according to the displayed function prototype. The new function is added to the function library you selected.

Note: If more than one scenario uses a function with the same name from different function libraries, the recovery process may fail. In this case, information regarding the recovery failure is displayed during the run session.

Click **Next**. The Recovery Operations Screen (described on page 1352) reopens, showing the function operation that you defined.

Post-Recovery Test Run Options Screen

When you clear the **Add another recovery operation** check box in the Recovery Operations Screen (described on page 1352) and click **Next**, the Post-Recovery Test Run Options screen opens. Post-recovery test run options specify how to continue the run session after QuickTest has identified the event and performed all of the specified recovery operations.



QuickTest can perform one of the following run session options after it performs the recovery operations you defined:

➤ **Repeat current step and continue**

The current step is the step that QuickTest was running when the recovery scenario was triggered. If you are using the **On error** activation option for recovery scenarios, the step that returns the error is often one or more steps later than the step that caused the trigger event to occur.

Thus, in most cases, repeating the current step does not repeat the trigger event. For more information, see “Enabling and Disabling Recovery Scenarios” on page 1377.

➤ **Proceed to next step**

Skips the step that QuickTest was running when the recovery scenario was triggered. Keep in mind that skipping a step that performs operations on your application may cause subsequent steps to fail.

➤ **Proceed to next action or component iteration**

Stops performing steps in the current action or component iteration and begins the next iteration from the beginning (or from the next action or component if no additional iterations of the current action or component are required).

➤ **Proceed to next test iteration**

Stops performing steps in the current action and begins the next QuickTest test iteration from the beginning (or stops running the test if no additional iterations of the test are required).

➤ **Restart current test run**

Stops performing steps and re-runs the test from the beginning.

➤ **Stop the test run**

Stops running the test.

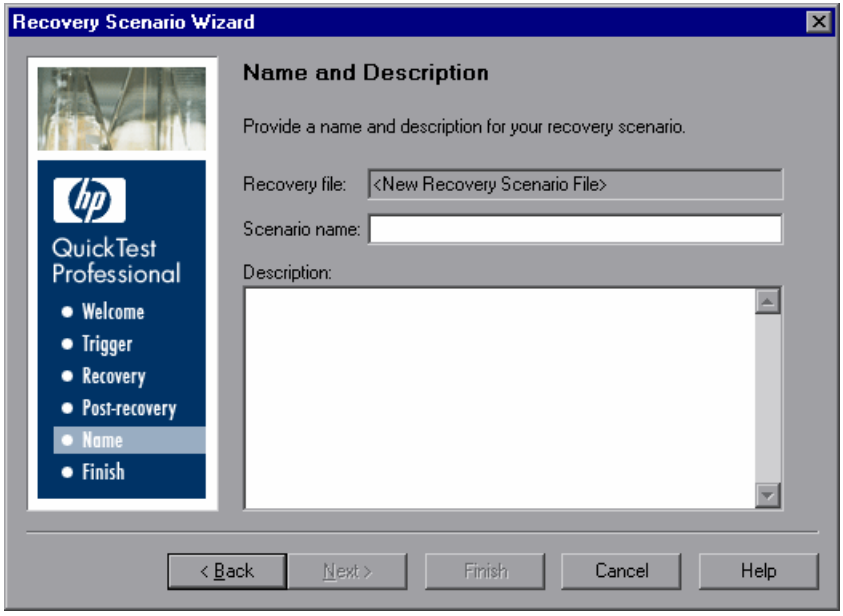
Note: If you chose **Restart Microsoft Windows** as a recovery operation, you can choose from only the last two test run options listed above.

Select a test run option and click **Next** to continue to the Name and Description Screen (described on page 1363).

Name and Description Screen

After you specify a test run option in the Post-Recovery Test Run Options Screen (described on page 1361), and click **Next**, the Name and Description screen opens.

In the Name and Description screen, you specify a name by which to identify your recovery scenario. You can also add descriptive information regarding the scenario.

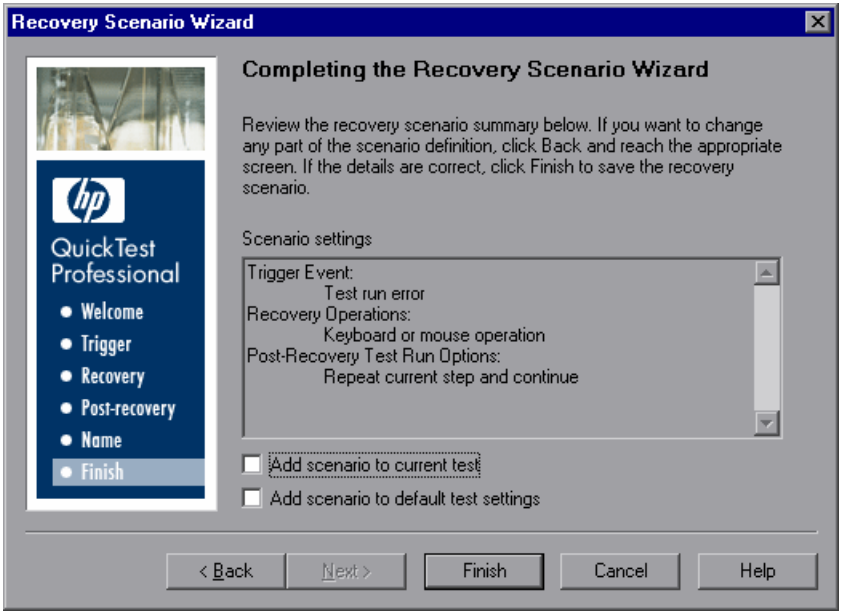


Enter a name and a textual description for your recovery scenario, and click **Next** to continue to the Completing the Recovery Scenario Wizard Screen (described on page 1364).

Completing the Recovery Scenario Wizard Screen

After you specify a recovery scenario name and description in the Name and Description Screen (described on page 1363) and click **Next**, the Completing the Recovery Scenario Wizard screen opens.

In the Completing the Recovery Scenario Wizard screen, you can review a summary of the scenario settings you defined. You can also specify whether to automatically associate the recovery scenario with the current test and/or to add it to the default settings for all new tests.



You can select the **Add scenario to current test** check box to associate this recovery scenario with the current test. When you click **Finish**, QuickTest adds the recovery scenario to the **Scenarios** list in the Recovery pane of the Test Settings dialog box.

You can select the **Add scenario to default test settings** check box to make this recovery scenario a default scenario for all new tests. The next time you create a test, this scenario will be listed in the **Scenarios** list in the Recovery pane of the Test Settings dialog box.

Note: You can remove scenarios from the default scenarios list. For more information, see "Defining Recovery Scenario Settings for Your Test" on page 1291.

Click **Finish** to complete the recovery scenario definition.

Saving the Recovery Scenario in a Recovery File

After you create or modify a recovery scenario in a recovery file using the Recovery Scenario Wizard, you need to save the recovery file.

Tip: If you have not yet saved the recovery file, and you click the **Close** button in the Recovery Scenario Manager dialog box, QuickTest prompts you to save the recovery file. Click **Yes**, and proceed with step 2 below. If you added or modified scenarios in an existing recovery file, and you click **Yes** to the message prompt, the recovery file and its scenarios are saved.

To save a new or modified recovery file:

1 In the Recovery Scenario Manager dialog box, click the **Save** button. If you added or modified scenarios in an existing recovery file, the recovery file and its scenarios are saved. If you are using a new recovery file, the Save Recovery Scenario dialog box opens.

Tip: You can also click the arrow to the right of the **Save** button and select **Save As** to save the recovery file under a different name.

---

2 In the sidebar, select the location in which you want to save the file, for example, File System or Quality Center Test Resources.

3 Browse to and select the folder in which you want to save the file.

4 In the **File name** box, enter a name for the file and click **Save**.

Tip: If you want to save the file as an attachment to a test in the Test Plan module in Quality Center, select **Quality Center Test Plan** in the sidebar, browse to and double-click the test, and then click **Save**.

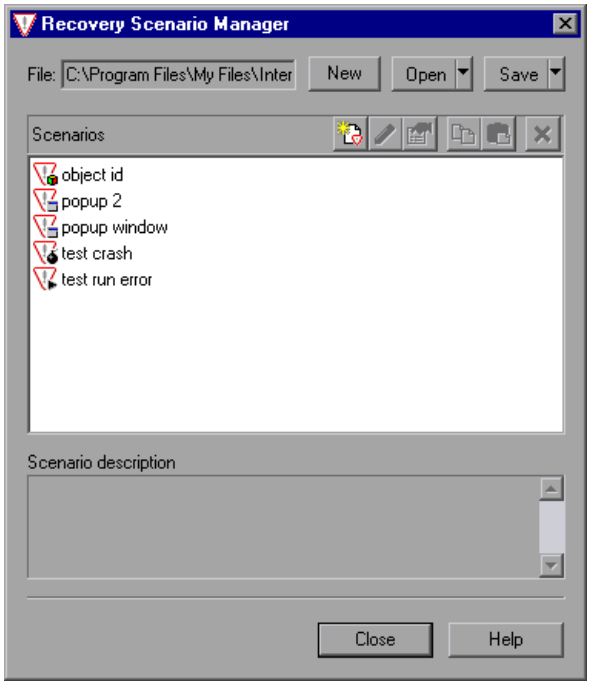
Note: When you specify a path to a resource in the file system or in Quality Center 9.x, QuickTest checks if the path, or a part of the path, exists in the Folders pane of the Options dialog box (**Tools** > **Options** > **Folders** node). If the path exists, you are prompted to define the path using only the relative part of the path you entered. If the path does not exist, you are prompted to add the resource's location path to the Folders pane and define the path relatively. For more information, see "Using Relative Paths in QuickTest" on page 316.

If you are working with the Resources and Dependencies model with Quality Center 10.00, you should specify an absolute Quality Center path. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.

The recovery file is saved in the specified location with the **.qrs** file extension.

Managing Recovery Scenarios

After you create recovery scenarios, you can use the Recovery Scenario Manager to manage them.



The Recovery Scenario Manager contains the following recovery scenario icons:

Icon	Description
	Indicates that the recovery scenario is triggered when a window pops up in an open application during the run session.
	Indicates that the recovery scenario is triggered when the property values of an object in an application match specified values.
	Indicates that the recovery scenario is triggered when a step in the test does not run successfully.
	Indicates that the recovery scenario is triggered when an open application fails during the run session.

The Recovery Scenario Manager enables you to manage existing scenarios by:

- Viewing Recovery Scenario Properties
- Modifying Recovery Scenarios
- Deleting Recovery Scenarios
- Copying Recovery Scenarios between Recovery Scenario Files

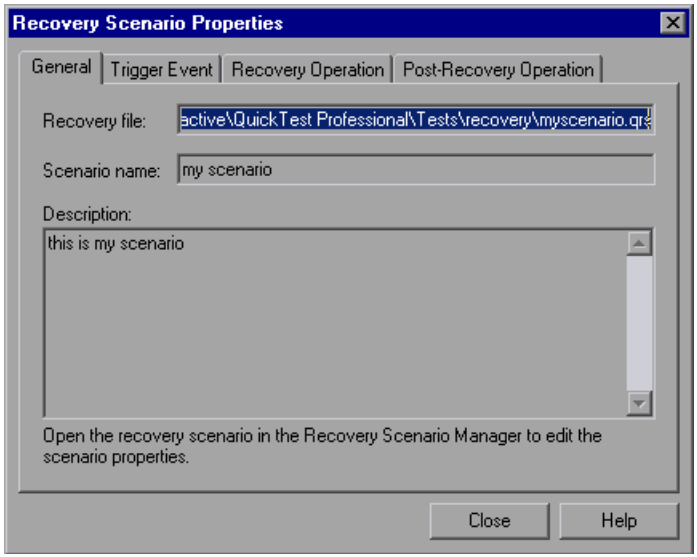
Viewing Recovery Scenario Properties

You can view properties for any defined recovery scenario.

To view recovery scenario properties:

1 In the **Scenarios** box, select the recovery scenario whose properties you want to view.

2 Click the **Properties** button. Alternatively, you can double-click a scenario in the **Scenarios** box. The Recovery Scenario Properties dialog box opens.



The Recovery Scenario Properties dialog box displays the following read-only information about the selected scenario:

- **General tab.** Displays the name and description defined for the recovery scenario, plus the name and path of the recovery file in which the scenario is saved.
- **Trigger Event tab.** Displays the settings for the trigger event defined for the recovery scenario.
- **Recovery Operation tab.** Displays the recovery operations defined for the recovery scenario.
- **Post-Recovery Operation tab.** Displays the post-recovery operation defined for the recovery scenario.

Modifying Recovery Scenarios

You can modify the settings for an existing recovery scenario.

To modify a recovery scenario:

1 In the **Scenarios** box, select the scenario that you want to modify.

2 Click the **Edit** button. The Recovery Scenario Wizard opens, with the settings you defined for the selected recovery scenario.

3 Navigate through the Recovery Scenario Wizard and modify the details as needed. For information on the Recovery Scenario Wizard options, see "Defining Recovery Scenarios" on page 1333.

Note: Modifications you make are not saved until you click **Save** in the Recovery Scenario Manager dialog box. If you have not yet saved your modifications, and you click the **Close** button in the Recovery Scenario Manager dialog box, QuickTest prompts you to save the recovery file. Click **Yes** to save your changes.

Deleting Recovery Scenarios

You can delete an existing recovery scenario if you no longer need it. When you delete a recovery scenario from the Recovery Scenario Manager, the corresponding information is deleted from the recovery scenario file.

Note: If a deleted recovery scenario is associated with a test, QuickTest ignores it during the run session.

To delete a recovery scenario:

1 In the **Scenarios** box, select the scenario that you want to delete.

2 Click the **Delete** button. The recovery scenario is deleted from the Recovery Scenario Manager dialog box.

Note: The scenario is not actually deleted until you click **Save** in the Recovery Scenario Manager dialog box. If you have not yet saved the deletion, and you click the **Close** button in the Recovery Scenario Manager dialog box, QuickTest prompts you to save the recovery file. Click **Yes** to save the recovery scenario file and delete the scenarios.

Copying Recovery Scenarios between Recovery Scenario Files

You can copy recovery scenarios from one recovery scenario file to another.

To copy a recovery scenario from one recovery scenario file to another:

1 In the **Scenarios** box, select the recovery scenario that you want to copy.

2 Click the **Copy** button. The scenario is copied to the Clipboard.

3 Click the **Open** button and select the recovery scenario file to which you want to copy the scenario, or click the **New** button to create a new recovery scenario file in which to copy the scenario.

4 Click the **Paste** button. The scenario is copied to the new recovery scenario file.

Notes:

- If a scenario with the same name already exists in the recovery scenario file, you can choose whether you want to replace it with the new scenario you have just copied.
- Modifications you make are not saved until you click **Save** in the Recovery Scenario Manager dialog box. If you have not yet saved your modifications, and you click the **Close** button in the Recovery Scenario Manager dialog box, QuickTest prompts you to save the recovery file. Click **Yes** to save your changes.

Associating Recovery Scenarios with Your Tests

After you create recovery scenarios, you associate them with selected tests so that QuickTest will perform the appropriate scenarios during the run sessions if a trigger event occurs. You can prioritize the scenarios and set the order in which QuickTest applies the scenarios during the run session. You can also choose to disable specific scenarios, or all scenarios, that are associated with a test. You can also define which recovery scenarios will be used as the default scenarios for all new tests.

Note: You can associate, remove, enable, disable, prioritize, and view the properties of the recovery scenarios associated with your test in the Resources pane. For more information, see "The Resources Pane" on page 1161.

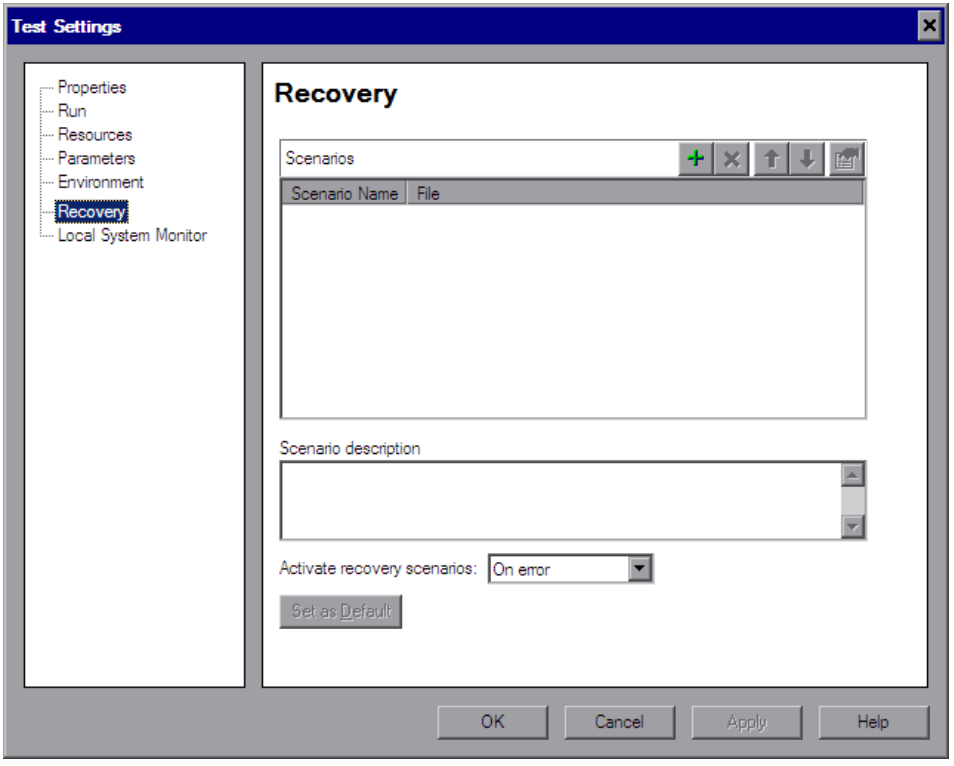
Adding Recovery Scenarios to Your Test

After you have created recovery scenarios, you can associate one or more scenarios with a test to instruct QuickTest to perform the recovery scenarios during the run session if a trigger event occurs. The Recovery pane of the Test Settings dialog box lists all the recovery scenarios associated with the current test.

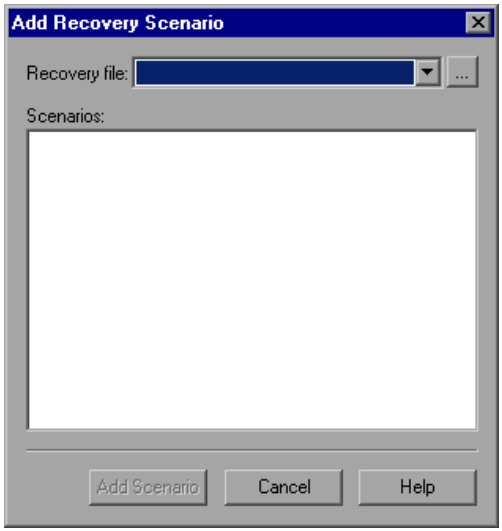
Tip: When a trigger event occurs, QuickTest checks for applicable recovery scenarios in the order in which they are displayed in the Recovery pane. You can change this order as described in "Setting Recovery Scenario Priorities" on page 1376.

To add a recovery scenario to a test:

1 Select **File** > **Settings**. The Test Settings dialog box opens. Select the **Recovery** node.



2 Click the **Add** button. The Add Recovery Scenario dialog box opens.



3 In the **Recovery file** box, select the recovery file containing the recovery scenarios you want to associate with the test. Alternatively, click the browse button to navigate to the recovery file you want to select. The **Scenarios** box displays the names of the scenarios saved in the selected file.

4 In the **Scenarios** box, select the scenarios that you want to associate with the test and click **Add Scenario**. The Add Recovery Scenario dialog box closes and the selected scenarios are added to the **Scenarios** list in the Recovery pane.

Tip: You can edit a recovery scenario file path by clicking the path once to highlight it, and then clicking it again to enter edit mode. For example, you may want to modify an absolute file path to be a relative file path. If you modify a recovery scenario file path, you must ensure that the recovery scenario is defined in the new path location before running your test.

Viewing Recovery Scenario Properties

You can view properties for any recovery scenario associated with your test.

Note: You modify recovery scenario settings from the Recovery Scenario Manager dialog box. For more information, see "Modifying Recovery Scenarios" on page 1370.

To view recovery scenario properties:

1 In the **Scenarios** box, select the recovery scenario whose properties you want to view.

2 Click the **Properties** button. Alternatively, you can double-click a scenario in the **Scenarios** box. The Recovery Scenario Properties dialog box opens, displaying read-only information regarding the settings for the selected scenario. For more information, see "Viewing Recovery Scenario Properties" on page 1368.

Setting Recovery Scenario Priorities

You can specify the order in which QuickTest performs associated scenarios during a run session. When a trigger event occurs, QuickTest checks for applicable recovery scenarios in the order in which they are displayed in the Recovery pane of the Test Settings dialog box.

To set recovery scenario priorities:

1 In the **Scenarios** box, select the scenario whose priority you want to change.

2 Click the **Up** or **Down** button. The selected scenario's priority changes according to your selection.

Removing Recovery Scenarios from Your Test

You can remove the association between a specific scenario and a test using the Recovery pane of the Test Settings dialog box. After you remove a scenario from a test, the scenario itself still exists, but QuickTest will no longer perform the scenario during a run session.

To remove a recovery scenario from your test:

1 In the **Scenarios** box, select the scenario you want to remove.

2 Click the **Remove** button. The selected scenario is no longer associated with the test.

Enabling and Disabling Recovery Scenarios

You can enable or disable specific scenarios and determine when QuickTest activates the recovery scenario mechanism in the Recovery pane of the Test Settings dialog box. When you disable a specific scenario, it remains associated with the test, but is not performed by QuickTest during the run session. You can enable the scenario at a later time.

You can also specify the conditions for which the recovery scenario is to be activated.

To enable/disable specific recovery scenarios:

➤ Select the check box to the left of one or more individual scenarios to enable them.

➤ Clear the check box to the left of one or more individual scenarios to disable them.

To define when the recovery mechanism is activated:

Select one of the following options in the **Activate recovery scenarios** box:

➤ **On every step.** The recovery mechanism is activated after every step. Note that choosing **On every step** may result in slower performance during the run session.

➤ **On error.** The recovery mechanism is activated only after steps that return an error return value.

Note that the step that returns an error is often not the same as the step that causes the exception event to occur.

For example, a step that selects a check box may cause a pop-up dialog box to open. Although the pop-up dialog box is defined as a trigger event, QuickTest moves to the next step because it successfully performed the check box selection step. The next several steps could potentially perform checkpoints, functions or other conditional or looping statements that do not require performing operations on your application. It may only be ten statements later that a step instructs QuickTest to perform an operation on the application that it cannot perform due to the pop-up dialog box. In this case, it is this tenth step that returns an error and triggers the recovery mechanism to close the dialog box. After the recovery operation is completed, the current step is this tenth step, and not the step that caused the trigger event.

➤ **Never.** The recovery mechanism is disabled.

Tip: You can also enable or disable specific scenarios or all scenarios associated with a test programmatically during the run session. For more information, see "Programmatically Controlling the Recovery Mechanism" on page 1379.

Setting Default Recovery Scenario Settings for All New Tests

You can click the **Set as Default** button in the Recovery pane of the Test Settings dialog box to set the current list of recovery scenarios to be the default scenarios for all new tests. Any future changes you make to the current recovery scenario list only affect the current test, and do not change the default list that you defined.

Programmatically Controlling the Recovery Mechanism

You can use the Recovery object to control the recovery mechanism programmatically during the run session. For example, you can enable or disable the entire recovery mechanism or specific recovery scenarios for certain parts of a run session, retrieve status information about specific recovery scenarios, and explicitly activate the recovery mechanism at a certain point in the run session.

By default, QuickTest checks for recovery triggers when an error is returned during the run session. You can use the Recovery object's Activate method to force QuickTest to check for triggers after a specific step in the run session. For example, suppose you know that an object property checkpoint will fail if certain processes are open when the checkpoint is performed. You want to be sure that the pass or fail of the checkpoint is not affected by these open processes, which may indicate a different problem with your application.

However, a failed checkpoint does not result in a run error. So by default, the recovery mechanism would not be activated by the object state. You can define a recovery scenario that looks for and closes specified open processes when an object's properties have a certain state. This state shows the object's property values as they would be if the problematic processes were open. You can instruct QuickTest to activate the recovery mechanism if the checkpoint fails so that QuickTest will check for and close any problematic open processes and then try to perform the checkpoint again. This ensures that when the checkpoint is performed the second time it is not affected by the open processes.

For more information on the Recovery object and its methods, see the *HP QuickTest Professional Object Model Reference*.

49

Working with the QuickTest Script Editor

The QuickTest Script Editor is a tool that enables you to open and edit multiple test scripts and function libraries simultaneously.

This chapter includes:

About the QuickTest Script Editor

The QuickTest Script Editor enables you to open and modify the scripts of multiple tests and function libraries, simultaneously. You can also create new function libraries. You can modify the script of a test, but you cannot create new tests, associate or remove associated function libraries, or change information such as existing test names, test settings, parameterization, or Data Table values.

For more information, see:

- "Working with Tests" on page 1393
- "Working with Function Libraries" on page 1397

Important Considerations

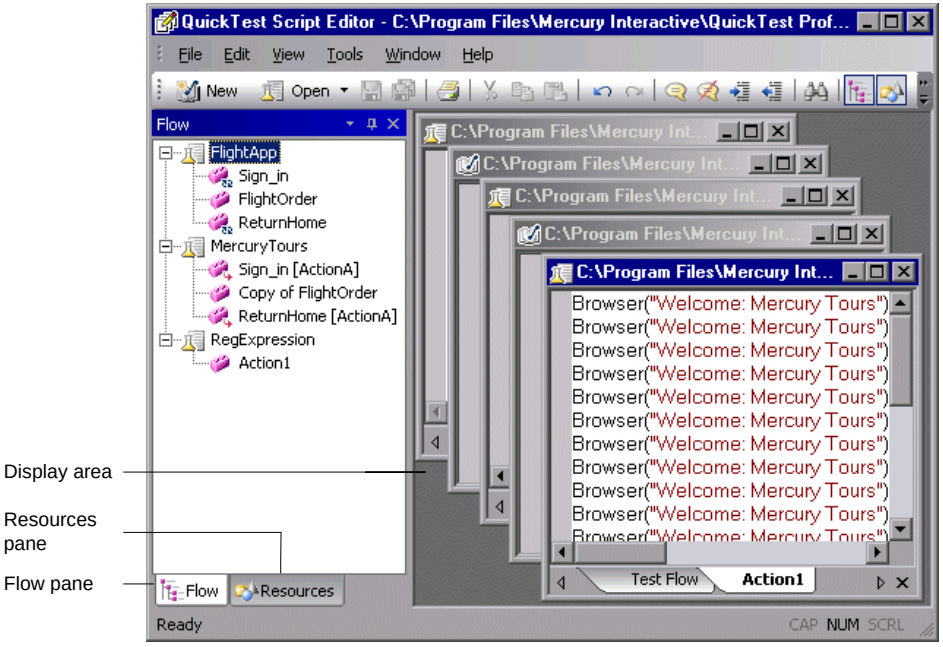
- The QuickTest Script Editor enables you to work with QuickTest tests and function libraries only. To work with components or scripted components, see Chapter 56, "Working with Business Process Testing."
- You cannot use the Script Editor to modify version control-enabled files stored in Quality Center, although you can open and view these files in read-only mode. If your tests and function libraries are part of a version control-enabled project in Quality Center, you must use QuickTest to modify these files.
- Tests created in earlier versions of QuickTest open in read-only mode.
  - If a test is stored in the file system or in Quality Center 9.x, you can update it to the current version by opening and saving it in QuickTest. After you save the test, you will not be able to open it in an earlier version of QuickTest or the Script Editor.
  - If a test is stored in Quality Center 10.00, the administrator must update it to the current version using the QuickTest Professional Asset Upgrade Tool for Quality Center, which upgrades all tests in the project simultaneously. After a test is upgraded, you cannot open it in an earlier version of QuickTest or the Script Editor. If the test is saved in a version-control-enabled project, the test opens only in read-only mode. To modify it, you must open it in QuickTest.

➤ The QuickTest Script Editor automatically adds a UTF-16 identifier to the start of each function library file that you save (either new or existing).

Understanding the QuickTest Script Editor Window

You open the QuickTest Script Editor by choosing **Start** > **Programs** > **QuickTest Professional** > **Tools** > **QuickTest Script Editor**.

An example of the QuickTest Script Editor window is shown below:



The QuickTest Script Editor window contains the following key elements:

➤ **Flow Pane.** Displays the flow of the action calls for each of the open tests.

➤ **Resources Pane.** Displays the open tests, its local actions and any function libraries associated with each test, as well as a list of all currently open function libraries.

➤ **Display area.** Displays a window for each of the open tests and function libraries.

For more information, see:

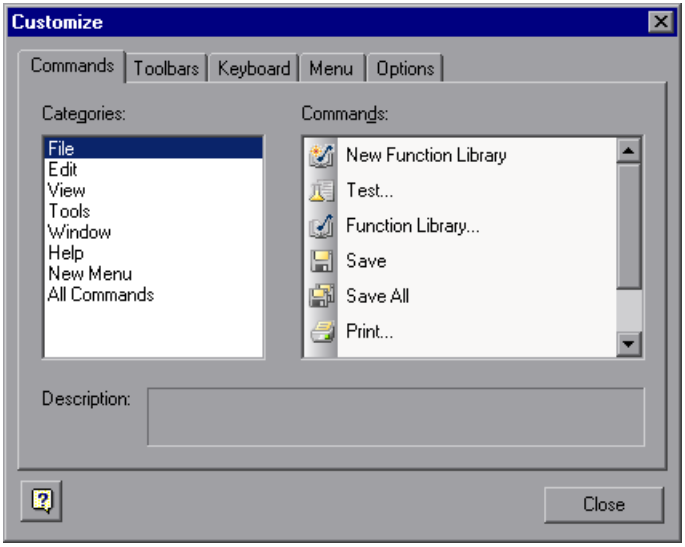
- "Customizing the QuickTest Script Editor Window" on page 1384
- "Understanding the Flow Pane" on page 1386
- "Understanding the Resources Pane" on page 1388
- "Understanding the Display Area" on page 1391

Customizing the QuickTest Script Editor Window

In the Customize dialog box, you can customize Script Editor toolbars, menus, and other display options in a similar way to many other Windows applications.

To open the Customize dialog box:

Right-click in the toolbar or menu bar and select **Customize**.



Click a tab and customize the Script Editor according to your requirements.

Commands Tab

You can add and move buttons and commands in the Script Editor toolbars and menus. You can also remove buttons and commands from the displayed toolbars and menus.

Toolbars Tab

You can select which of the available toolbars to display in the Script Editor window. You can choose whether to display text labels for the toolbar buttons. You can also reset the toolbar display to the default.

Keyboard Tab

You can assign new keyboard shortcuts for toolbar and menu commands, or modify and remove existing shortcuts. You can also reset all of the keyboard shortcuts to the default.

Menu Tab

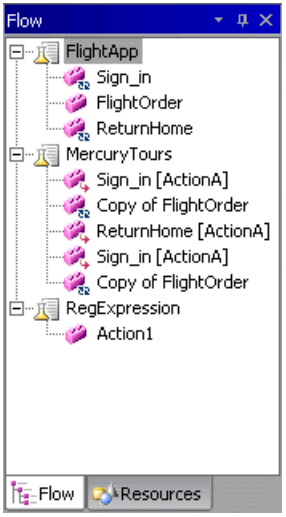
You can select which of the available menus to display in the Script Editor window, and the commands that appear in the context menus. You can choose how the menus are animated, and whether they are displayed with a shadow. You can also reset the displayed menus to the default.

Options Tab

You can select whether to show tooltips for toolbar buttons, whether to show shortcut keys in the tooltips, and whether to display toolbar buttons as large or small icons.

Understanding the Flow Pane

The Flow pane displays the test flow (action call flow) for each currently open test. Each open test is displayed as a node in a tree, and each node contains the hierarchy of all the actions that were called in the test, including calls to local, reusable, and external actions. You can also see each test's action calls in the Flow pane of the relevant test window in the display area.



The Flow pane displays the following icons:

Option	Description
	A test
	A call to a local action
	A call to an external action
	A call to a reusable action
	A call to an action whose path is not saved with the test
	A looped action call, meaning a call to an action that was already called earlier in the test flow hierarchy

You can perform the following operations in the Flow pane:

- **Display the script of an action.** Double-click the action, or right-click the action and select **Show**. Each shown action is displayed as a tab in its test window. If you show an external action, the test containing the called action is added to the tree in the Flow pane and Resources pane, and the selected action is displayed in a new test window in the display area.
- **Display the line in a test script that calls a selected action.** Right-click the action and select **Go to Action Call**. The action call script line is highlighted in the relevant action tab of the test window.
- **Display the test or action properties.** Right-click the test or action and then select **Properties**. The name of the test or action and its path are displayed. If it was defined with a relative path in QuickTest, then the path is displayed as .\<name of action or function library>. If the action is an external action, the **External** check box is selected.
- **Close a test.** Right-click the test and then select **Close**. If you have any unsaved changes, you are prompted to save them.

If a test contains a call to an action that does not exist, or cannot be found, the action still appears in the tree in the Flow pane. An error message stating that the action cannot be found is displayed when you try to show the action.

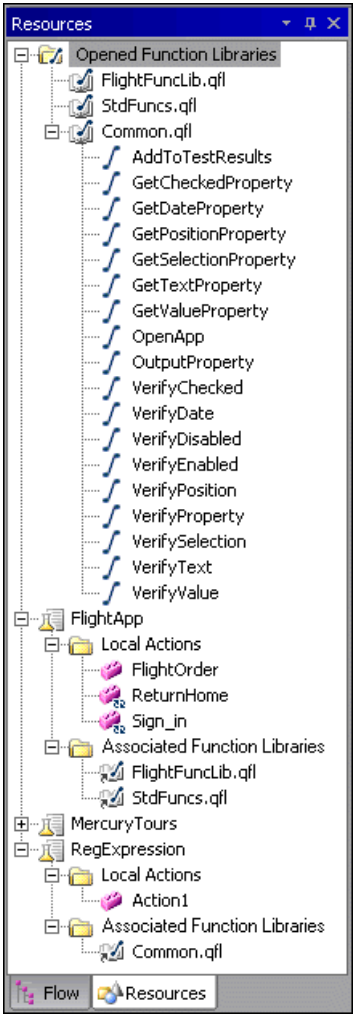
Tips:

- You can right-click in the Flow pane title bar to view available display options and decide how to display the Flow pane. For example, you can auto hide the pane, dock it, or close it.
- 
You can click the **Toggle Flow View** toolbar button to hide or show the Flow pane view.

For more information, see “Working with Tests” on page 1393.

Understanding the Resources Pane

The Resources pane displays all the currently open tests and their resources (actions and associated function libraries). Each test is displayed as a node in the tree, and each node contains the actions and function libraries associated with the test. All currently open function libraries, and their functions, are also displayed in a separate node at the top of the pane.



The Resources pane displays the following icons:

Option	Description
	An open function library
	A public function defined in a function library
	A private function defined in a function library
	A test
	A local action
	A reusable action
	A link to a function library that is associated with a test

You can perform the following operations in the Resources pane:

- **Display the script of a local action or the code of a function library.** Double-click the action or function library, or right-click and select **Show**. Each shown action is displayed as a tab in its test window, and each function library is displayed in a separate window.
- **Display the properties of local actions or function libraries.** Right-click the action or function library and select **Properties**. The name of the action or function library, and its path are displayed. If it was defined with a relative path in QuickTest, then the path is displayed as .\<name of action or function library>. If the action is a reusable action, the **Reusable** check box is selected.
- **Display the location of a function in a function library.** Right-click the function in the **Opened Function Libraries** folder, and select **Go to Function Definition**. The first line of the function definition is highlighted in the function library window.
- **Close a function library.** Right-click the function library in the **Opened Function Libraries** folder and select **Close**. If you have any unsaved changes, you are prompted to save them.
- **Close a test.** Right-click the test and select **Close**. If you have any unsaved changes, you are prompted to save them.

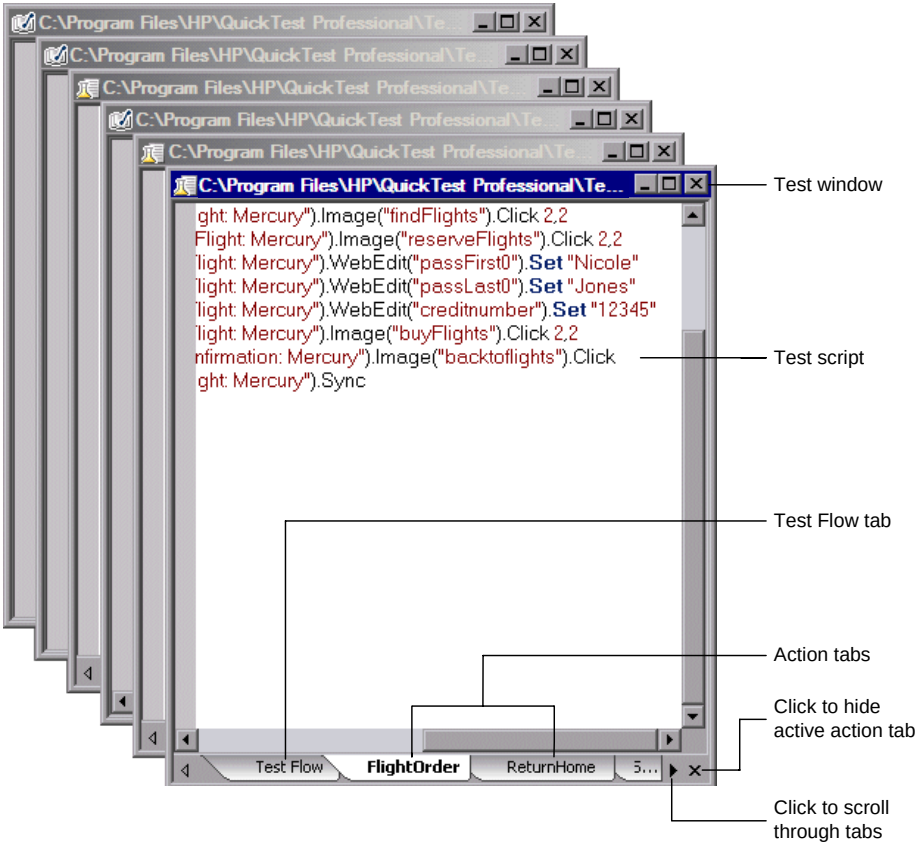
Tips:

- You can right-click in the Resources pane title bar to view available display options and decide how to display the Resources pane. For example, you can auto hide the pane, dock it, or close it.
-  You can click the **Toggle Resources View** toolbar button to hide or show the Resources pane view.

For more information, see “Working with Tests” on page 1393, and “Working with Function Libraries” on page 1397.

Understanding the Display Area

The display area contains a separate window for each open test or function library, and each test window contains a tab for each local action open in the test.



Tip: You can use the options in the **Windows** menu to decide how these windows are arranged in the display area.

To display an action or function library in the display area, either double-click the action or function library in the Flow or Resources pane, or right-click and then select **Show**. In the test windows, a tab is displayed for each open local action. If you double-click a call to an external action in the Flow or Resources pane, the test containing the called action is displayed in the tree in the Flow pane and Resources pane, and the test is displayed as a new window in the display area, with a tab for the called action. (If the test containing the action is already open, the tab for the called action is added to the test window if it is not already shown.)

If an action does not exist, or cannot be found, a message is displayed when you try to open it.

Not all the available tabs for open local actions may be visible at the bottom of the test window. You can navigate between the available tabs by clicking the arrows at the bottom of the window to scroll through the tabs.

You can select an action tab and then click the **Hide Action** button at the bottom right of the window to remove the action's tab from the window. Note that the action is not closed, only hidden, and therefore you will not be prompted to save any changes made.

To display the line of a test script that calls a selected action, right-click the action in the tree in the Flow pane, and then select **Go to Action Call**. The action call script line is highlighted in the relevant action tab of the test window.

To display the location of a function in a function library, right-click the function in the **Opened Function Libraries** folder at the top of the tree in the Resources pane, and then select **Go to Function Definition**. The first line of the function definition is highlighted in the function library window.

You can use the Editor Options dialog box (**Tools > View Options**) to customize how test scripts and function libraries are displayed in the QuickTest Script Editor. For example, you can choose whether to display line numbers, or change the font and color used to display the scripts. For more information on using the Editor Options dialog box, see Chapter 30, "Customizing the Expert View and Function Library Windows."

For more information, see "Working with Tests" on page 1393, and "Working with Function Libraries" on page 1397.

Working with Tests

You can open multiple existing tests, edit them and then save them.

You can also customize the way the test scripts are displayed, find and replace text strings within each test, and print the tests. For more information, see:

- "Customizing the Expert View and Function Library Windows" on page 895
- "Finding Text Strings" on page 847
- "Replacing Text Strings" on page 849
- "Printing a Test" on page 332

Opening Tests

You can open tests from the file system and tests that are saved in a Quality Center project. You can open as many tests as you want. When you open a test, it is displayed in the tree, and the Flow pane of the test window lists the calls to all of the top-level actions in the test.

Tip: You can open an existing test by dragging it from the file system (Windows Explorer) to the Script Editor window. You can open a recently used test by selecting it from the **Recent Files** list in the **File** menu.

To open a test:

1 (Optional) Click the **Quality Center Connection** button and connect to Quality Center, if required. For more information on connecting to Quality Center, see "Connecting to and Disconnecting from Quality Center" on page 1418.

2 Open the test in one of the following ways:

- Click the **Open Test** toolbar button, or select **File** > **Open** > **Test**. The Open Test dialog box opens. In the sidebar, select the location of the test, for example, File System or Quality Center Test Plan. Browse to and select the test and click **Open**.
- In the Flow pane, double-click the test to open it, or right-click the test, and select **Show**.

  If you only want to view the test script and not modify it, you can select the **Open in read-only mode** check box at the bottom of the dialog box.

Note: The **Open** button toggles between **Open Test** and **Open Function Library**, according to the active window in the display area. To change the **Open Function Library** button to **Open Test**, click the drop-down arrow next to the button and then select **Test**, or click a test window in the display area.

The <path of test> window opens in the display area, listing the action calls in the test.

The test and all its actions are displayed in the tree in the Flow pane, and the local actions and function libraries are displayed in the tree in the Resources pane.

Note: The test opens in read-only mode if:

- The test you select is currently open by another user. In this case, you are notified that the test is already open, and by whom.
- The test was created in an earlier version of QuickTest. For more information, see "Important Considerations" on page 1382.
- The test is open in QuickTest, and you try to open the same test in the QuickTest Script Editor on the same computer, or vice versa.

In addition, if you open a test in the QuickTest Script Editor, it is locked, and no other users can modify it until you close it.

Editing Tests

You can use QuickTest Script Editor to edit multiple test scripts simultaneously. You edit the tests by adding or modifying information, copying and pasting, or dragging and dropping information from other tests and function libraries.

When working with tests in the QuickTest Script Editor, you cannot create new tests, or save existing tests with a new name. You can modify only the test script. This means that you cannot change information such as test settings, parameterization, Data Table values, and so on.

When you modify a test script or function library, make sure that you make all changes in the test script using the correct syntax, format, and spelling because the QuickTest Script Editor does not do this for you.

To edit a test:

1. (Optional) Click the **Quality Center Connection** button and connect to Quality Center, if required. For more information on connecting to Quality Center, see "Connecting to and Disconnecting from Quality Center" on page 1418.
2. Open the tests to be edited, as well as those from which you want to copy information, if required. You can also open any function libraries that you may need.

3 Edit the tests as required. An asterisk (*) is displayed in the title bar of the edited test windows until you save your changes.

Tips:

➤ You can change selected commented text to uncommented text, or vice versa, by using the **Comment Block** or **Uncomment Block** toolbar buttons or by using the **Edit** menu options.

➤ You can indent or outdent selected text by using the **Indent** or **Outdent** toolbar buttons or by using the **Edit** menu options.

Saving Tests

You can save the active test, or all the open tests and function libraries.

To save a test:

➤ Click the **Save** toolbar button or select **File** > **Save** to save the active test. (The active test is the test window that is currently in focus in the display area.)

➤ Click the **Save All** toolbar button or select **File** > **Save All** to save all the open tests and function libraries. If any open function library was not previously saved, the Save Function Library dialog box opens. For more information, see “Saving Function Libraries” on page 1401.

Closing Tests

You can close a test from the Flow pane, the Resources pane, or from the display area.

To close a test:

In the Flow pane or Resources pane, right-click the test you want to close and select **Close**, or in the display area, click the **Close** button at the top of the test window you want to close. The test window is closed, and the test is removed from the Flow and Resource panes.

Note: If you have unsaved changes, you are prompted to save these changes before closing the test.

Working with Function Libraries

Function library files can contain VBScript functions, subroutines, classes, modules, and so forth, which you can associate with your test to provide additional functionality. Using the QuickTest Script Editor, you can open and edit multiple function libraries, and create new function libraries.

You can customize the way the function library code is displayed, find and replace text strings within each function library, and print the function libraries. For more information, see:

- "Customizing the Expert View and Function Library Windows" on page 895
- "Finding Text Strings" on page 847
- "Replacing Text Strings" on page 849
- "Printing a Function Library" on page 917

Opening Function Libraries

You can open function libraries from the file system and function libraries that are part of a Quality Center project. You can open as many function libraries as you want. The QuickTest Script Editor works with **.qfl**, **.vbs**, and **.txt** function library files.

After you open a function library, it is displayed in a function library window in the display area, and the function library and its functions are displayed in the **Opened Function Libraries** folder at the top of the tree in the Resources pane. If the function library is associated with an open test, it is also displayed under the test as a function library link in the **Associated Function Libraries** folder in the tree in the Resources pane.

Tip: You can open an existing function library by dragging it from the file system (Windows Explorer) to the Script Editor window. You can open a recently used function library by selecting it from the **Recent Files** list in the **File** menu.

To open a function library:

1 (Optional) Click the **Quality Center Connection** button and connect to Quality Center, if required. For more information on connecting to Quality Center, see "Connecting to and Disconnecting from Quality Center" on page 1418.

2 Open the function library in one of the following ways:

- In the Resources pane, double-click the function library to open, or right-click the function library, and select **Open Function Library**.
- Click the **Open Function Library** toolbar button, select **File** > **Open** > **Function Library**. The Open Function Library dialog box opens. In the sidebar, select the location of the function library, for example, File System or Quality Center Test Plan, and browse to and select a function library. Click **Open**.

Note: The **Open** button toggles between **Open Test** and **Open Function Library**, according to the active window in the display area. To change the **Open Test** button to **Open Function Library**, click the arrow next to the button and then select **Function Library**, or click a function library window in the display area.

The <function library path> window opens, and the function library is displayed in the **Opened Function Libraries** folder at the top of the tree in the Resources pane.

If you open a function library from the file system that is opened by another user, you are notified if changes are made by the other user, and given the option to accept or reject the changes made.

If you open a function library saved in Quality Center, the file is locked by you. No other user can modify it until you close it.

Note: The function library opens in read-only mode if:

- The function library you select is currently open by another user. In this case, you are notified that the function library is already open, and by whom.
- The function library was created in an earlier version of QuickTest. For more information, see "Important Considerations" on page 1382.
- The function library is open in QuickTest, and you try to open the same test in the QuickTest Script Editor on the same computer, or vice versa.

Creating Function Libraries

The Script Editor enables you to create new function libraries. These can be associated with tests using QuickTest.

To create a function library:

1. Click the **New Function Library** toolbar button, or select **File** > **New Function Library**. A function library window opens in the display area. By default, the name of the function library is **Library<number>**.
2. Enter the required code for the function library.
3. Save the function library as described in "Saving Function Libraries" on page 1401.

Editing Function Libraries

You can edit the function code of multiple function libraries. You edit the function libraries by adding or modifying information, copying and pasting, or dragging and dropping information from other function libraries and tests. Function libraries that open in read-only mode cannot be edited.

To edit a function library:

1. Open the function libraries to be edited, as well as those from which you want to copy information, if required. You can also open any tests you may need.
2. Edit the function libraries as required. An asterisk (*) is displayed in the title bar of the edited function library windows until you save your changes.

Tips:

- You can change selected commented text to uncommented text, or vice versa, by using the **Comment Block** or **Uncomment Block** toolbar buttons or by using the **Edit** menu options.

- You can indent or outdent selected text by using the **Indent** or **Outdent** toolbar buttons or by using the **Edit** menu options.

Saving Function Libraries

You can save the active function library, rename and save the function library to a different location, or save all open function libraries and tests. You can save the function library in the file system or in Quality Center.

To save a function library:

1 (Optional) Connect to a non-version-control-enabled project on a Quality Center server. For more information, see "Connecting to and Disconnecting from Quality Center" on page 1418.

2 Click the **Save** toolbar button or select **File** > **Save** to save the active function library. Note that the active function library is the function library window that is currently in focus in the display area.

- Select **File** > **Save As** to rename the active function library or to save it to a new location.

- Click the **Save All** toolbar button or select **File** > **Save All** to save all the open function libraries and tests.

The Save Function Library dialog box opens.

3 In the sidebar, select the location to save the test, for example, File System or Quality Center Test Resources.

Note: If you are connected to a Quality Center project with version control enabled, you cannot save the function library to Quality Center. Therefore, only the **File System** button is displayed in the sidebar.

---

4 In the **File name** box, enter a name and file extension for the function library. The QuickTest Script Editor works with **.qfl**, **.vbs**, and **.txt** function library files.

The file name must not begin with a space or contain any of the following characters: \ / : * ? " < > | % '

If you save the function library to Quality Center, the file path must not contain two consecutive semicolons (;;).

5 Click **Save**. The function library is saved to the specified location.

Closing Function Libraries

You can close a function library from the Resources pane, or from the display area.

To close a function library:

In the Resources pane, right-click the function library (in the **Opened Function Libraries** folder) you want to close and select **Close**, or in the display area, click the **Close** button at the top of the function library window you want to close. The function library window is closed, and the function library is removed from the **Opened Function Libraries** folder in the Resources pane.

Note: If you have unsaved changes, you are prompted to save these changes before closing the function library.

50

Automating QuickTest Operations

Just as you use QuickTest to automate the testing of your applications, you can use the QuickTest Professional automation object model to automate your QuickTest operations. Using the objects, methods, and properties exposed by the QuickTest automation object model, you can write scripts that configure QuickTest options and run tests instead of performing these operations manually using the QuickTest interface.

Automation scripts are especially useful for performing the same tasks multiple times or on multiple tests, or quickly configuring QuickTest according to your needs for a particular environment or application.

This chapter includes:

About Automating QuickTest Operations

You can use the QuickTest Professional automation object model to write scripts that automate your QuickTest operations. The QuickTest automation object model provides objects, methods, and properties that enable you to control QuickTest from another application.

What is Automation?

Automation is a Microsoft technology that makes it possible to access software objects inside one application from other applications. These objects can be created and manipulated using a scripting or programming language such as VBScript or VC++. Automation enables you to control the functionality of an application programmatically.

An **object model** is a structural representation of software objects (classes) that comprise the implementation of a system or application. An object model defines a set of classes and interfaces, together with their properties, methods and events, and their relationships.

What is the QuickTest Automation Object Model?

Essentially all configuration and run functionality provided via the QuickTest interface is in some way represented in the QuickTest automation object model via objects, methods, and properties. Although a one-on-one comparison cannot always be made, most dialog boxes in QuickTest have a corresponding automation object, most options in dialog boxes can be set and/or retrieved using the corresponding object property, and most menu commands and other operations have corresponding automation methods.

You can use the objects, methods, and properties exposed by the QuickTest automation object model, along with standard programming elements such as loops and conditional statements to design your script.

Automation scripts are especially useful for performing the same tasks multiple times or on multiple tests, or quickly configuring QuickTest according to your needs for a particular environment or application.

For example, you can create and run an automation script from Microsoft Visual Basic that loads the required add-ins for a test, starts QuickTest in visible mode, opens the test, configures settings that correspond to those in the Options, Test Settings, and Record and Run Settings dialog boxes, runs the test, and saves the test.

You can then add a simple loop to your script so that your single script can perform the operations described above for multiple tests.

You can also create an initialization script that opens QuickTest with specific configuration settings. You can then instruct all of your testers to open QuickTest using this automation script to ensure that all of your testers are always working with the same configuration.

Deciding When to Use QuickTest Automation Scripts

Creating a useful QuickTest automation script requires planning, design time, and testing. You must always weigh the initial investment with the time and human-resource savings you gain from automating potentially long or tedious tasks.

Any QuickTest operation that you must perform many times in a row or must perform on a regular basis is a good candidate for a QuickTest automation script.

The following are just a few examples of useful QuickTest automation scripts:

- **Initialization scripts.** You can write a script that automatically starts QuickTest and configures the options and the settings required for testing a specific environment.
- **Maintaining your tests.** You can write a script that iterates over your collection of tests to accomplish a certain goal. For example:
  - **Updating values.** You can write a script that opens each test with the proper add-ins, runs it in update run mode against an updated application, and saves it when you want to update the values in all of your tests to match the updated values in your application.
  - **Applying new options to existing tests.** When you upgrade to a new version of QuickTest, you may find that the new version offers certain options that you want to apply to your existing tests. You can write a script that opens each existing test, sets values for the new options, then saves and closes it.
  - **Modifying Actions and Action Parameters.** You can retrieve the entire contents of an action script, and add a required step, such as a call to a new action. You can also retrieve the set of action parameters for an action and add, remove, or modify the values of action parameters.
- **Calling QuickTest from other applications.** You can design your own applications with options or controls that run QuickTest automation scripts. For example, you could create a Web form or simple Windows interface from which a product manager could schedule QuickTest runs, even if the manager is not familiar with QuickTest.

Choosing a Language and Development Environment for Designing and Running Automation Scripts

You can choose from a number of object-oriented programming languages for your automation scripts. For each language, there are a number of development environments available for designing and running your automation scripts.

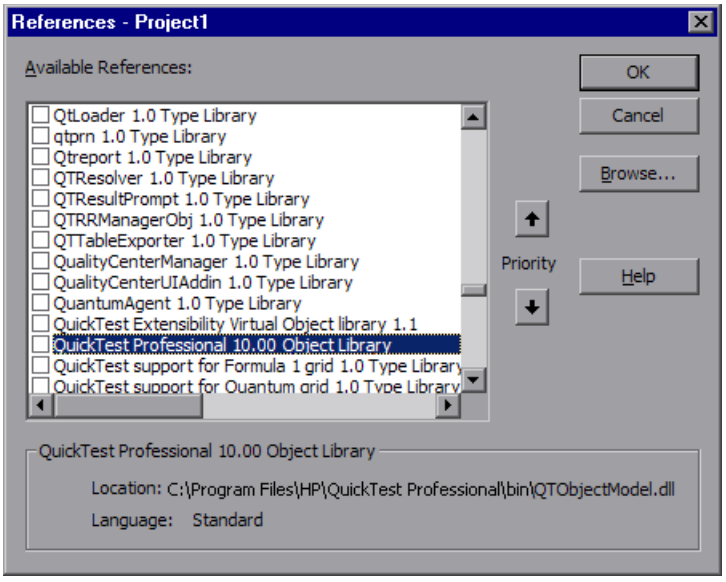
Writing Your Automation Script

You can write your QuickTest automation scripts in any language and development environment that supports automation. For example, you can use: VBScript, JavaScript, Visual Basic, Visual C++, or Visual Studio .NET.

Some development environments support referencing a type library. A **type library** is a binary file containing the description of the objects, interfaces, and other definitions of an object model.

If you choose a development environment that supports referencing a type library, you can take advantage of features like Microsoft IntelliSense, automatic statement completion, and status bar help tips while writing your script. The QuickTest automation object model supplies a type library file named **QTObjectModel.dll**. This file is stored in **<QuickTest installation folder>\bin**.

If you choose an environment that supports it, be sure to reference the QuickTest type library before you begin writing or running your automation script. For example, if you are working in Microsoft Visual Basic, select **Project** > **References** to open the References dialog box for your project. Then select **QuickTest Professional <Version> Object Library** (where <**Version**> is the current installed version of the QuickTest automation type library).



Running Your Automation Script

There are several applications available for running automation scripts. You can also run automation scripts from the command line using Microsoft's Windows Script Host.

For example, you could use the following command line to run your automation script:

```
WScript.exe /E:VBSCRIPT myScript.vbs
```

Learning the Basic Elements of a QuickTest Automation Script

Like most automation object models, the root object of the QuickTest automation object model is the **Application** object. The Application object represents the application level of QuickTest. You can use this object to return other elements of QuickTest such as the Test object (which represents a test document), Options object (which represents the Options dialog box), or Addins collection (which represents a set of add-ins from the Add-in Manager dialog box), and to perform operations like loading add-ins, starting QuickTest, opening and saving tests, and closing QuickTest.

Each object returned by the Application object can return other objects, perform operations related to the object and retrieve and/or set properties associated with that object.

Every automation script begins with the creation of the QuickTest Application object. Creating this object does not start QuickTest. It simply provides an object from which you can access all other objects, methods and properties of the QuickTest automation object model.

Note: You can also optionally specify a remote QuickTest computer on which to create the object (the computer on which to run the script). For more information, see **Running Automation Programs on a Remote Computer** in the **Introduction** section of the *QuickTest Automation Object Model Reference* in the *QuickTest Professional Help*.

The structure for the rest of your script depends on the goals of the script. You may perform a few operations before you start QuickTest such as retrieving the associated add-ins for a test, loading add-ins, and instructing QuickTest to open in visible mode.

After you perform these preparatory steps, if QuickTest is not already open on the computer, you can open QuickTest using the Application.Launch method. Most operations in your automation script are performed after the Launch method.

For information on the operations you can perform in an automation program, see the online *HP QuickTest Professional Object Model Reference*. For more information on this Help file, see "Using the QuickTest Automation Reference" on page 1411.

When you finish performing the necessary operations, or you want to perform operations that require closing and restarting QuickTest, such as changing the set of loaded add-ins, use the Application.Quit method.

Generating Automation Scripts

The Properties pane of the Test Settings dialog box, the General pane of the Options dialog box, and the Object Identification dialog box each contain a **Generate Script** button. Clicking this button generates an automation script file (**.vbs**) containing the current settings from the corresponding dialog box.

You can run the generated script as is to open QuickTest with the exact configuration of the QuickTest application that generated the script, or you can copy and paste selected lines from the generated files into your own automation script.

For example, the generated script for the Options dialog box may look something like this:

```
Dim App 'As Application
Set App = CreateObject("QuickTest.Application")
App.Launch
App.Visible = True
App.Options.DisableVORecognition = False
App.Options.AutoGenerateWith = False
App.Options.WithGenerationLevel = 2
App.Options.TimeToActivateWinAfterPoint = 500
...
...
App.Options.WindowsApps.NonUniqueListItemRecordMode = "ByName"
App.Options.WindowsApps.RecordOwnerDrawnButtonAs = "PushButtons"
App.Folders.RemoveAll
```

For more information on the **Generate Script** button and for information on the options available in the Options, Object Identification, and Test Settings dialog boxes, see Chapter 4, "Configuring Object Identification", Chapter 44, "Setting Global Testing Options", and Chapter 45, "Setting Options for Individual Tests."

Using the QuickTest Automation Reference

The QuickTest Automation Object Model Reference is a Help file that provides detailed descriptions, syntax information, and examples for the objects, methods, and properties in the QuickTest automation object model.

You can open the *HP QuickTest Professional Automation Object Model Reference* from:

- QuickTest program folder (**Start** > **Programs** > **QuickTest Professional** > **Documentation** > **QuickTest Automation Reference**)
- Main QuickTest Help (**Help** > **QuickTest Professional Help** > **HP QuickTest Professional Advanced References** > **HP QuickTest Professional Automation Object Model**)

Part XI

Working with Quality Center

51

Integrating with Quality Center

To ensure comprehensive testing of your application or applications, you typically must create and run many tests. HP Quality Center, the centralized quality solution, can help you organize and control the testing process.

Note: References to Quality Center features and options in this chapter apply to all currently supported versions of Quality Center, unless otherwise noted. However, they may not be supported in the Quality Center edition you are using.

For a list of the supported versions of Quality Center, see the *HP QuickTest Professional Readme*.

For more information on Quality Center editions, see the *HP Quality Center User Guide*.

This chapter includes:

About Working with Quality Center

QuickTest integrates with Quality Center, the HP centralized quality solution. Quality Center helps you maintain a project of all kinds of tests (such as QuickTest tests, business process tests, manual tests, tests created using other HP products, and so on) that cover all aspects of your application's functionality. Each test in your project is designed to fulfill a specified testing requirement of your application. To meet the goals of a project, you organize the tests in your project into unique groups.

Quality Center provides an intuitive and efficient method for scheduling and running tests, collecting results, analyzing the results, and managing test versions. It also features a system for tracking defects, enabling you to monitor defects closely from initial detection until resolution.

A Quality Center project is a database for collecting and storing data relevant to a testing process. For QuickTest to access a Quality Center project, you must connect to the local or remote Web server where Quality Center is installed. When QuickTest is connected to Quality Center, you can create tests and save them in your Quality Center project. After you run your tests, you can view the results in Quality Center.

You must have the following access permissions to use QuickTest with Quality Center:

- Full read and write permissions to the Quality Center cache folder (located on the Quality Center client side)
- Full read and write permissions to the QuickTest Add-in for Quality Center installation folder

---

Tip: For information about the various QuickTest add-ins, see the *HP QuickTest Professional Add-ins Guide*.

When working with Quality Center, you can associate tests with external files stored in the Test Resources module of a Quality Center project. You can associate external files for all tests or for a single test. For example, suppose you set the shared object repository mode as the default mode for new tests. You can instruct QuickTest to use a specific object repository stored in Quality Center.

For more information on specifying external files for all tests, see Chapter 44, "Setting Global Testing Options." For more information on specifying external files for a single test, see Chapter 45, "Setting Options for Individual Tests."

You can report defects to a Quality Center project either automatically as they occur, or manually directly from the QuickTest Test Results window. For information on manually or automatically reporting defects to a Quality Center project, see "Submitting Defects Detected During a Run Session" on page 1013.

For more information on working with Quality Center, see the *HP Quality Center User Guide*. For the latest information and tips regarding QuickTest and Quality Center integration, see the *HP QuickTest Professional Readme* (available from **Start** > **Programs** > **QuickTest Professional** > **Readme**).

Connecting to and Disconnecting from Quality Center

If you are working with both QuickTest and Quality Center, QuickTest can communicate with your Quality Center project.

You can connect or disconnect QuickTest to or from a Quality Center project at any time during the testing process. However, do not disconnect QuickTest from Quality Center while a QuickTest test is opened from Quality Center or while QuickTest is using a shared resource from Quality Center (such as a shared object repository or Data Table file).

Note: You can connect to any currently supported version of Quality Center. See the *HP QuickTest Professional Readme* for a list of the supported versions of Quality Center. For more information, see "Quality Center Connectivity Add-in" on page 1424.

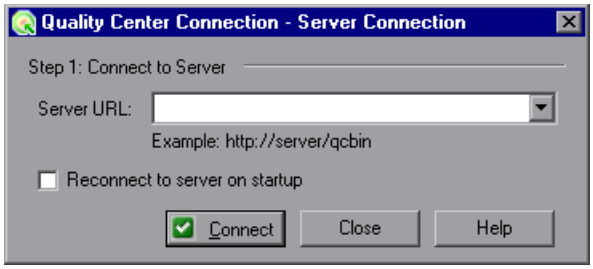
Connecting QuickTest to Quality Center

The connection process has two stages. First, you connect QuickTest to a local or remote Quality Center server. This server handles the connections between QuickTest and the Quality Center project.

Next, you log in and choose the project you want QuickTest to access. The project stores tests and run session information for the application you are testing. Note that Quality Center projects are password protected, so you must provide a user name and a password.

To connect QuickTest to a Quality Center server:

1 Select **File** > **Quality Center Connection** or click the **Quality Center Connection** toolbar button. The Quality Center Connection - Server Connection dialog box opens.



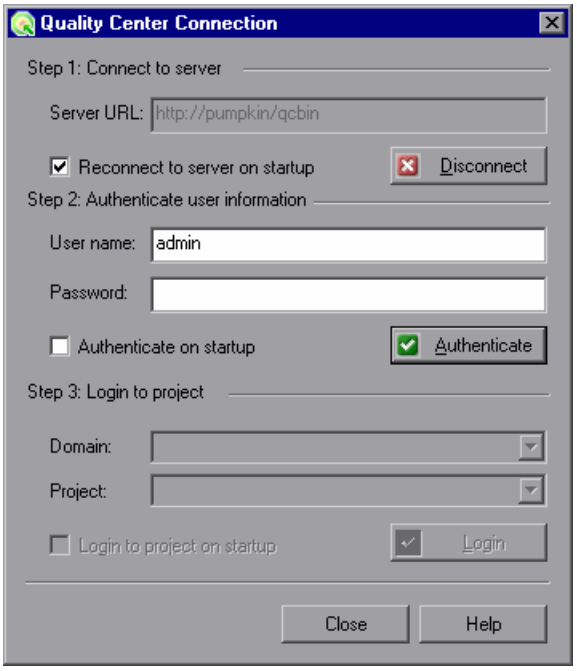
2 In the **Server URL** box, type the URL address of the Web server where Quality Center is installed.

Note: You can choose a Quality Center server accessible via a Local Area Network (LAN) or a Wide Area Network (WAN).

---

3 To automatically reconnect to the Quality Center server the next time you open QuickTest, select the **Reconnect to server on startup** check box.

4 Click **Connect**. The Quality Center Connection dialog box opens.



The Quality Center server name is displayed in read-only format in the Server URL box.

5 In the **User name** box, type your Quality Center user name.

6 In the **Password** box, type your Quality Center password.

7 Click **Authenticate** to authenticate your user information against the Quality Center server.

After your user information has been authenticated, the edit boxes in the **Authenticate user information** area are displayed in read-only format. The **Authenticate** button changes to a **Change User** button.

Tip: You can log in to the same Quality Center server using a different user name by clicking **Change User**, and then entering a new user name and password and clicking **Authenticate** again.

---

8 In the **Domain** box, select the domain that contains the Quality Center project. Only those domains that you have permission to connect to are displayed.

9 In the **Project** box, select the project with which you want to work. Only those projects for which you are a defined user are displayed.

10 Click **Login**.

11 To automatically reconnect to the Quality Center server the next time you open QuickTest, select the **Reconnect to server on startup** check box.

12 If the **Reconnect to server on startup** check box is selected, then the **Authenticate on startup** check box is enabled. To automatically authenticate your user information the next time you open QuickTest, select the **Authenticate on startup** check box.

13 If the **Authenticate on startup** check box is selected, the **Login to project on startup** check box is enabled. To log in to the selected project on startup, select the **Login to project on startup** check box.

14 Click **Close** to close the Quality Center Connection dialog box. The Quality Center icon is displayed on the status bar to indicate that QuickTest is currently connected to a Quality Center project.

Tip: To view the current Quality Center connection, point to the **Quality Center** icon on the status bar. A tooltip displays the Quality Center server name and project to which QuickTest is connected. To open the Quality Center Connection dialog box, double-click the **Quality Center** icon.

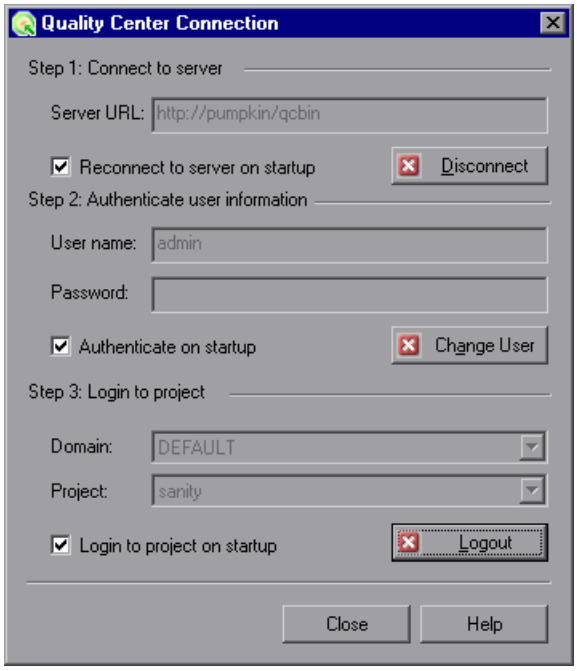
Disconnecting QuickTest from Quality Center

You can disconnect QuickTest from a Quality Center project or from a Quality Center server at any time. Note that if you disconnect QuickTest from a Quality Center server without first disconnecting from a project, the QuickTest connection to that project database is automatically disconnected.

Note: If a Quality Center test, or shared file (such as a shared object repository or Data Table file) is open when you disconnect from Quality Center, then QuickTest closes it.

To disconnect QuickTest from a Quality Center server:

1 Select **File** > **Quality Center Connection** or click the **Quality Center Connection** toolbar button. The Quality Center Connection dialog box opens.



2 To disconnect QuickTest from the selected project, in the **Step 3: Login to project** area, click **Logout**.

3 To disconnect QuickTest from the selected Quality Center server, in the **Step 1: Connect to server** area, click **Disconnect**.

Tip: You can log in to the same Quality Center server using a different user name by clicking **Change User** and then entering a new user name and password and clicking **Authenticate** again.

---

4 Click **Close** to close the Quality Center Connection dialog box.

Integrating QuickTest with Quality Center

Integrating QuickTest with Quality Center enables you to store and access files in a Quality Center project, as well as use the QCUtil object to access the wide range of functionality provided in the Quality Center Open Test Architecture API.

Quality Center Connectivity Add-in

You integrate QuickTest with Quality Center using the Quality Center Connectivity Add-in. This add-in is installed on your QuickTest computer automatically when you connect QuickTest to Quality Center using the Quality Center Connection dialog box. You can also install it manually from the Quality Center Add-ins page by choosing **Help** > **Add-ins Page** > **HP Quality Center Connectivity** in Quality Center.

To view the version of the Quality Center Connectivity Add-in that is currently installed on your computer, select **Help > About** and then click the **Product Information** button. For more information, see "Viewing Product Information" on page 73.

Integrating with Quality Center

At its most basic level, integrating QuickTest with Quality Center enables you to store and access QuickTest tests and resource files in a Quality Center project, when QuickTest is connected to Quality Center.

You can take advantage of all of the features provided with the Resources and Dependencies model. For information, see "Using the Resources and Dependencies Model" on page 1447.

In addition, your tests and function libraries can use the QCUtil object to access and use the full functionality of the Quality Center OTA (Open Test Architecture). This enables you to automate integration operations during a run session, such as reporting a defect directly to a Quality Center database. For more information, see the **Utility** section of the *HP QuickTest Professional Object Model Reference* and the Quality Center Open Test Architecture documentation.

You can also use the TDOTA object in your QuickTest automation scripts to access the Quality Center OTA. For more information, see the *QuickTest Professional Automation Object Model Reference* (**Help** > **QuickTest Professional Help** > **HP QuickTest Professional Advanced References** > **HP QuickTest Professional Automation Object Model**).

Saving Tests to a Quality Center Project

When QuickTest is connected to a Quality Center project, you can create new tests in QuickTest and save them directly to your project. To save a test, you give it a descriptive name and associate it with the relevant subject in the test plan tree. This helps you to keep track of the tests created for each subject and to quickly view the progress of test planning and creation.

Tip: If you later need to save standalone, portable copies of tests stored in a Quality Center project, you can do so. For example, you may need to open or run a test while travelling because you do not have access to Quality Center. For more information, see "Creating Portable Copies of Your Tests" on page 326.

To save a test to a Quality Center 10.00 project:

1. Connect to a Quality Center server and project. For more information, see "Connecting QuickTest to Quality Center" on page 1418.

2. In QuickTest, click **Save** or select **File** > **Save** to save the test. The Save Test dialog box opens.
3. Click **Quality Center Tests** in the sidebar and browse to and select the relevant subject folder.

4 In the **File Name** box, enter a name for the test. Use a descriptive name that will help you easily identify the test. A test name:

- ➤ Cannot exceed 220 characters (including the path)
- ➤ Cannot begin or end with spaces
- ➤ Cannot include the following characters: \ / : * ? " < > | % '
- ➤ Cannot contain two consecutive semicolons (;;)

5 Confirm that the **Save Active Screen files** is selected if you want to save the Active Screen files with your test. If you clear this box, your Active Screen files will be deleted, and you will not be able to edit your test using Active Screen options. For more information, see "Saving a Test" on page 324.

6 Click **Save** to save the test and close the dialog box. Note that the text in the status bar changes while QuickTest saves the test.

The next time you start Quality Center 10.00, the new test will be included in the Quality Center test plan tree. For more information, see the *HP Quality Center User Guide*.

Opening Tests from a Quality Center Project

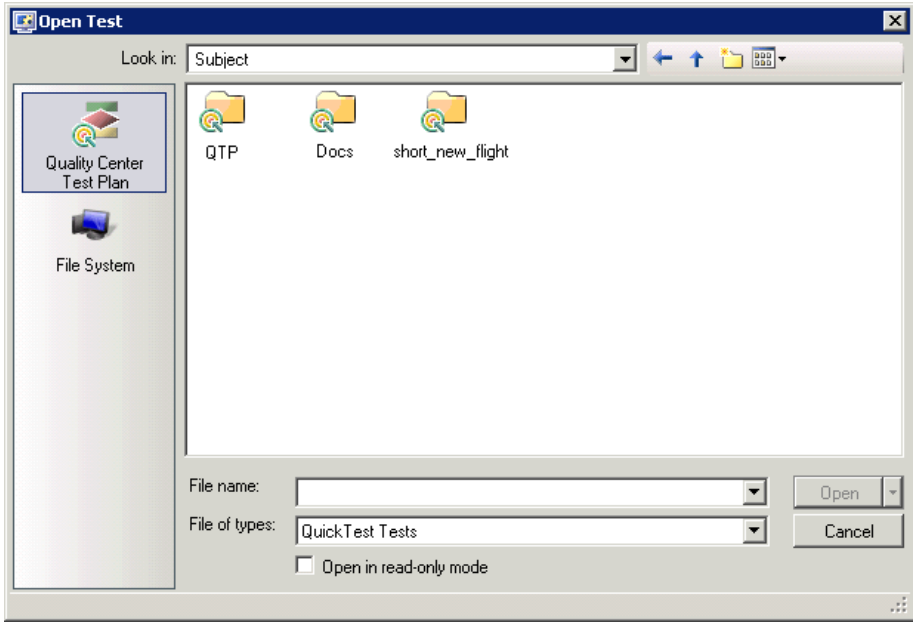
When QuickTest is connected to a Quality Center project, you can open QuickTest tests that are a part of your Quality Center project. You locate tests according to their position in the test plan tree, rather than by their actual location in the file system. You can also open tests from the recent tests list in the **File** menu.

Note: If a test is stored in Quality Center and was created using an earlier version, it opens in read-only mode. To edit the test, it must be upgraded to the current version using the QuickTest Professional Asset Upgrade Tool for Quality Center.

To open a test from a Quality Center 10.00 project:

1 Connect to a Quality Center server and project. For more information, see "Connecting QuickTest to Quality Center" on page 1418.

2 In QuickTest, click **Open** or select **File** > **Open** > **Test** to open the test. The Open Test dialog box opens.



3 Click the **Quality Center Test Plan** button in the sidebar, and then browse to and select the required test. The test is displayed in the **File name** box.

Note: If the test is stored in a Quality Center project with version control support, you can view version control information for the test by clicking the **Views** down arrow and selecting **Details**.

- The **Name** column lists the names of the tests that belong to the selected subject.
- The **Modified By** column indicates the Quality Center user that created or last modified the test.
- The **Checked Out To** column indicates the Quality Center user to whom the test is currently checked out. If the test is checked in, this is blank.

---

4 If you want to open the test in read-only mode, select the **Open in read-only mode** check box.

5 Click **Open** to open the test, or click the **Open** down arrow and select **Open and Check out** if the test is checked into a version-control-enabled project and you want to modify it after it opens.

As QuickTest downloads and opens the test, the operations it performs are displayed in the status bar.

When the test opens, the QuickTest title bar displays [Quality Center], the full subject path and the test name. For example:

[Quality Center] Subject\System\qa_test1

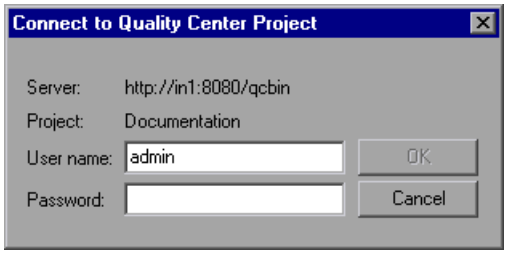
The test opens in read-only mode if:

- You selected **Open in read-only mode**
- You opened a test that is currently checked in to the Quality Center version control database (for projects that support version control)
- You opened a test that is currently checked out to another user (for projects that support version control)
- You opened a test from an earlier version of Quality Center, and the test has not yet been updated to the current format.

For more information, see "Opening Tests from a Quality Center Project with Version Control Support" on page 1429.

Opening Tests from the Recent Files List

You can open Quality Center tests from the recent files list in the **File** menu. If you select a test located in a Quality Center project, but QuickTest is currently not connected to Quality Center or to the correct project for the test, the Connect to Quality Center Project dialog box opens and displays the correct server, project, and the name of the user who most recently opened the test on this computer.



The Connect to Quality Center Project dialog box also opens if you choose to open a test that was last edited on your computer using a different Quality Center user name. You can either log in using the displayed name or you can click **Cancel** to stay logged in with your current user name.

Opening Tests from a Quality Center Project with Version Control Support

When you click the **Open** toolbar button or select **File** > **Open** > **Test** to open a test from a Quality Center project with version control support, and you display the **Details** view (by clicking the **Views** down arrow and selecting **Details** in the Open Test dialog box), the **Checked Out To** column specifies the user name of the Quality Center user to whom the test is checked out, if it is checked out.

When you open a test from a Quality Center project with version control support, the test opens in read-write or read-only mode depending on the current version control status of the test.

The table below summarizes the version control statuses and the open mode for each status:

Description	Open Mode
Checked in. If the test is currently checked in to the version control database, there is no indication in the dialog box.	Read-only
Checked out to you. If the test is currently checked out to you, your user name is displayed is the dialog box.	Read-write
Checked out to another user. If the test is currently checked out to another user, that user's name is displayed is the dialog box.	Read-only

For more information on working with tests stored in a Quality Center project with version control, see "Managing Versions of Assets in Quality Center" on page 1480.

Working with Template Tests

Template tests serve as the basis for all QuickTest tests created in Quality Center. A template test is a QuickTest test that contains default test settings. For example, a template test might specify the QuickTest add-ins, associated function libraries, and recovery scenarios that are associated with a test. You can modify these test settings in the Test Settings dialog box (**File > Settings**) in QuickTest.

In addition to default test settings, a template test can also contain any comments or steps you want to include with all new QuickTest tests created in Quality Center. For example, you may want to add a comment notifying users which add-ins are associated with the template test, or you may want to add a step that opens a specific Web page or application at the beginning of every test. Any steps or comments you add to a template test are included in all new tests created in Quality Center that are based on that template test.

A default template test is installed on each Quality Center client when the QuickTest Professional Add-in for Quality Center is installed. You can modify this default template test, or you can create customized template tests with various test settings.

All template tests are saved in your Quality Center project (except for the default template test, which is located on the Quality Center client) and do not need to be copied to each user's local computer. This enables users to customize their local default template tests, if needed, and still have access to globally maintained template tests. For more information, see "Working with New Template Tests" on page 1432.

When tests based on a specific template test are run from Quality Center, QuickTest automatically loads the associated add-ins and applies the required settings, as defined in the test.

Working with the Default Template Test

When you install the QuickTest Add-in for Quality Center, default template tests for all supported QuickTest versions are installed in the **<QuickTest Add-in for Quality Center folder>\bin\Templates** folder on your computer (for example: C:\Program Files\HP\QuickTest Add-in for Quality Center\bin\Templates\Template10).

When a Quality Center user creates a new QuickTest test in Quality Center, the default template test for the installed QuickTest version is automatically associated with the test unless the users selects another template test, as described in "Creating a QuickTest Test in Quality Center" on page 1434.

You can modify the template test that is installed by default with the QuickTest Add-in for Quality Center. Because the default template test is installed locally, any changes you make to the template test are applied only to tests created on your computer (using the Quality Center client).

Alternatively, you can create a new template test, as described in the following sections.

For more information on applying the default template test to a new test in Quality Center, see "Creating a QuickTest Test in Quality Center" on page 1434.

Working with New Template Tests

When you create new template tests, they are stored in your Quality Center project, making them available to all Quality Center users as the basis for new QuickTest tests created in that Quality Center project.

You can create multiple template tests, each for a specific testing purpose. For example, you may want to create one template test for QuickTest tests that test Web applications with ActiveX controls, and another for QuickTest tests that test standard Windows applications. You would associate the ActiveX and Web Add-ins with the first template test. For the second template test, you would not associate any QuickTest add-ins at all, but you might specify the Windows application that you want to test. You could also make other modifications to the test settings for each of the template tests, as needed.

As you create each template test, you can save it with a descriptive name that clearly indicates its purpose, such as, ActiveX_Web_Addins_Template or Std_Windows_Template_Test. Users can then choose the appropriate template test when creating QuickTest tests in Quality Center.

Note: When you define a template test that associates specific QuickTest add-ins, make sure that the add-ins are actually installed on the QuickTest computer on which the which the test will eventually run. Otherwise, when the test is run, QuickTest will not be able to load the required add-ins and the test may fail. For more information on running QuickTest tests from Quality Center, see the Quality Center documentation.

Creating a New Template Test

You create a template test by first creating a blank test in QuickTest with the required test settings. Then, in the **Test Plan** module of your Quality Center project, you browse to your QuickTest test and mark it as a **Template Test**.

When you save the test in QuickTest, you should apply a descriptive name that clearly indicates its purpose. For example, if the template test is to used to associate the ActiveX and Web Add-ins with a new test, you could call it ActiveX_Web_Addins_Template.

Tip: In the Quality Center test plan tree (Test Plan module), you may want to create a special folder for your template tests. This will enable other users to quickly locate the relevant template test when they create new QuickTest tests in Quality Center.

To create a template test:

1 In QuickTest:

a Open QuickTest with the required add-ins loaded. For more information on loading QuickTest add-ins, see the section on loading QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.

b Define the required settings in the Test Settings dialog box (**File > Settings**). For more information, see "Using the Test Settings Dialog Box" on page 1262.

c If you want to include comments or steps in all tests based on this template test, add them.

d Click the **Save** button or select **File > Save** to save the test. The Save Test to Quality Center dialog box opens. Save the test to your Quality Center project using a descriptive name that clearly indicates its purpose. For more information, see "Saving Tests to a Quality Center Project" on page 1425.

2 In Quality Center:

a Open the project in Quality Center, click the **Test Plan** button on the sidebar to open the Test Plan module, and browse to the test you saved in step d.

b Right-click the test and select **Mark as Template Test**. The test is converted to a template test.

3 Repeat steps 1 and 2 to create additional template tests, as needed.

Creating a QuickTest Test in Quality Center

In Quality Center, you create QuickTest tests in the Test Plan module. When you create a QuickTest test, you apply a template test to it. The template test applies pre-defined test settings to your new QuickTest test. For example, a template test can specify the QuickTest add-ins, function libraries, and recovery scenarios to be associated with your test. It can also include comments and steps (statements), as needed. You can choose either the default template test stored on your QuickTest client, or a template test that is saved in your Quality Center project.

If you do not have any template tests saved in your Quality Center project, or if you select **<None>** in the **Template** box (in the Create New Test dialog box shown on page 1426), Quality Center uses the settings defined in the template test that was installed with the **QuickTest Add-in for Quality Center** on your Quality Center client. For more information, see "Working with the Default Template Test" on page 1431. Otherwise, if you have at least one template test saved in your Quality Center project, you can select it when creating a new QuickTest test. For more information, see "Working with New Template Tests" on page 1432.

Note: When you create a QuickTest test in Quality Center, you must choose a template test that specifies the QuickTest add-ins to be associated with the test. Otherwise the required QuickTest add-ins will not be loaded during the run session.

Your new QuickTest test will use all of the settings defined in the template test you choose. When the test runs from Quality Center, QuickTest uses the settings specified in the Test Settings dialog box, and automatically loads the required QuickTest add-ins.

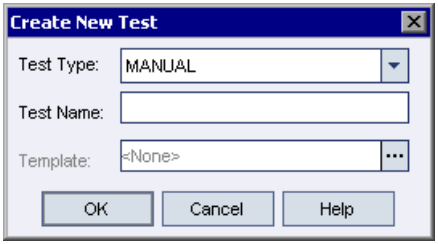
Note: The following procedure describes how to create a test in Quality Center using a template test. This procedure may be different depending on your version of Quality Center. For the most updated instructions on creating a new test in Quality Center, see the *HP Quality Center User Guide*.

To create a test in Quality Center using a template test:

1 In Quality Center, click the **Test Plan** button on the sidebar to open the Test Plan module.

2 In the test plan tree, select a folder.

3 Click the **New Test** button, or select **Test** > **New Test**. The Create New Test dialog box opens.



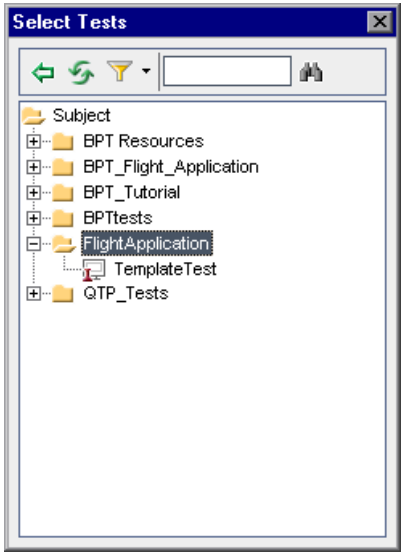
Note: The **Template** box is displayed only if the **QuickTest Professional Add-in for Quality Center** is installed on your computer. If the **Template** box is not displayed, you must install the **QuickTest Professional Add-in for Quality Center** from the QuickTest Professional DVD or from the More Quality Center Add-ins page (opened from the Quality Center Options window, or from **Help** > **Add-ins Page**).

4 From the **Test Type** list, select **QUICKTEST_TEST**.

5 In the **Test Name** box, type a name for the test start using English (Roman) letters, numbers, and underscores (if needed). Note that a test name cannot exceed 220 characters (including the path), cannot contain two consecutive semicolons (;;), cannot begin or end with spaces, and cannot include any of the following characters: \ / : * ? " < > | % '

6 Click the **Template** box browse button. The Select Tests dialog box opens.

7 Expand the folder containing your template test.



8 Select the template test on which to base your new test and click the **Add** button . The Select Tests dialog box closes and the template test you selected is displayed in the **Template** box (in the Create New Test dialog box).

9 In the Create New Test dialog box, click **OK**. The new test is created with the test settings defined in the template test.

10 The new test is displayed in the Test Plan tree under the subject folder you selected.

Note: If the Required Fields dialog box opens, set the required values and click **OK**. For more information, see the *HP Quality Center Administrator Guide*.

---

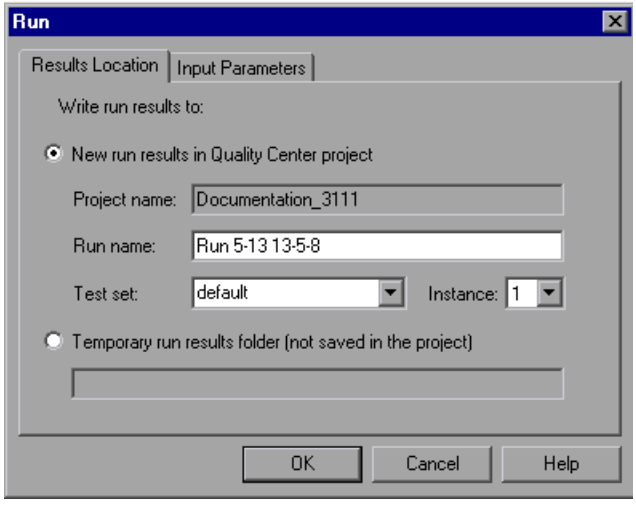
11 Continue creating the test. For more information on creating tests in Quality Center, see the *HP Quality Center User Guide*.

Running a Test Stored in a Quality Center Project from QuickTest

QuickTest can run a test from a Quality Center project and save the run results in the project. To save the run results, you specify a name for the run session and a test set in which to store the results.

To save run results to a Quality Center project:

1 In QuickTest, click the **Run** button or select **Automation** > **Run**. The Run dialog box opens.



2 The **Project name** box displays the Quality Center project to which you are currently connected.

To save the run results in the Quality Center project, accept the default **Run name**, or type a different one in the box.

3 Accept the default **Test set** or select a different one.

4 If there is more than one instance of the test in the test set, specify the instance of the test for which you want to save the results in the **Instance** box.

Note: A **test set** is a group of tests selected to achieve specific testing goals. For example, you can create a test set that tests the user interface of the application or the application's performance under stress. You define test sets when working in the Quality Center test run mode. For more information, see your Quality Center documentation.

To run the test, overwriting the previous test run results, select the **Temporary run results folder (not saved in the project)** option.

Note: QuickTest stores temporary test run results for all tests in **%TMP%\TempResults**. The path in the text box of the **Temporary run results folder (not saved in the project)** option is read-only and cannot be changed.

---

5 Click **OK**. The Run dialog box closes and QuickTest begins running the test. As QuickTest runs the test, it highlights each step in the Keyword View.

When the test stops running, the Test Results window opens unless you have cleared the **View results when test run ends** check box in the Run pane of the Options dialog box. For more information on the Options dialog box, see Chapter 44, "Setting Global Testing Options."

When the test stops running, **Uploading** is displayed in the status bar. The Test Results window opens when the uploading process is completed.

Note: You can report defects to a Quality Center project either automatically as they occur, or manually directly from the QuickTest Test Results window. For more information, see "Submitting Defects Detected During a Run Session" on page 1013.

Setting Preferences for Quality Center Test Runs

You can run QuickTest tests that are stored in a Quality Center database via QuickTest, via a Quality Center client that is installed on your computer, or via a remote Quality Center client. When Quality Center runs your QuickTest test, it uses the associated add-ins list to load the proper add-ins for your test on the QuickTest computer. For more information, see "Modifying Associated Add-Ins" on page 1268 and "Working with Template Tests" on page 1430.

Notes:

- You cannot run QuickTest tests from Quality Center if the QuickTest computer is logged off or locked.
- By default, QuickTest opens and runs in hidden mode when Quality Center activates it to run a test in a test set. This helps to improve performance. You can change this setting in the QuickTest Remote Agent. You can also instruct the Remote Agent to display a tooltip window indicating that QuickTest is running a Quality Center test in hidden mode. For more information, see "Setting QuickTest Remote Agent Preferences" on page 1441.

You can instruct QuickTest to report a defect for each failed step when Quality Center test runs on your QuickTest computer. You can also submit defects to Quality Center manually from the QuickTest Test Results window. For more information, see "Submitting Defects Detected During a Run Session" on page 1013.

Before you instruct a remote Quality Center client to run QuickTest tests on your computer, you must give Quality Center permission to use your QuickTest application. You can also view or modify the QuickTest Remote Agent settings.

Enabling Quality Center to Run Tests on a QuickTest Computer

For security reasons, remote access to your QuickTest application is not enabled. If you want to allow Quality Center (or other remote access clients) to open and run QuickTest tests, you must select the **Allow other HP products to run tests and components** option in the Options dialog box.

Note: If you want to run QuickTest tests remotely from Quality Center, and QuickTest is installed on Windows XP Service Pack 2, Windows 2003 Server, or Windows Vista, you must first change DCOM permissions and open firewall ports. For more information, see the *HP QuickTest Professional Installation Guide*.

In addition, if you want to run QuickTest tests remotely from Quality Center, and QuickTest is installed on Windows Vista, you must disable User Account Control (UAC) in **Windows Control Panel** > **User Accounts** before you first connect to Quality Center. For more information, see the *HP QuickTest Professional Installation Guide*.

To enable remote Quality Center clients to run tests on your QuickTest computer:

1 Open QuickTest.

2 Select **Tools** > **Options** or click the **Options** toolbar button . The Options dialog box opens.

3 Click the **Run** node.

4 Select the **Allow other HP products to run tests and components** check box.

For more information on this option, see "Setting Run Testing Options" on page 1253.

Tip: For full access to QuickTest tests from Quality Center, you must also have the QuickTest Add-in for Quality Center installed on your Quality Center client computer. This enables you to view the test and view the run results in the Test Results viewer. For more information on this add-in, go to the QuickTest Professional Add-in screen (accessible from the main Quality Center screen).

Setting QuickTest Remote Agent Preferences

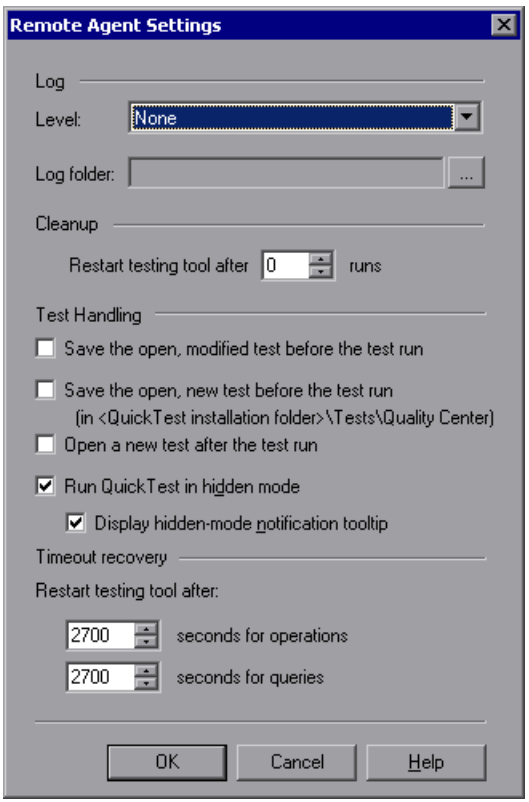
When you run a QuickTest test or business process test from Quality Center, the QuickTest Remote Agent opens on the QuickTest computer. The QuickTest Remote Agent determines how QuickTest behaves when a test is run by a remote application such as Quality Center.

You can open the Remote Agent Settings dialog box at any time to view or modify the settings that your QuickTest application uses when Quality Center runs a test on your computer.

To open the Remote Agent Settings dialog box:

1 Select **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Remote Agent**. The Remote Agent opens and the **Remote Agent** icon is displayed in the task bar tray.

2 Right-click the **Remote Agent** icon and select **Settings**. The Remote Agent Settings dialog box opens.



3 View or modify the settings in the dialog box. For more information, see “Understanding the Remote Agent Settings Dialog Box” on page 1443.

4 Click **OK** to save your settings and close the dialog box.

5 Right-click the **Remote Agent** icon and select **Exit** to end the Remote Agent session.

Understanding the Remote Agent Settings Dialog Box

The Remote Agent Settings dialog box enables you to view or modify the settings that QuickTest uses when Quality Center runs a QuickTest test or business process test on your computer.

The Remote Agent Settings dialog box contains the following options:

Option	Description
Level	The level of detail to include in the log that is created when Quality Center runs a QuickTest test or business process test. **None.** (Default) No log is created. **Low.** The log lists any Quality Center-QuickTest communication errors. **Medium.** The log includes Quality Center-QuickTest communication errors and information on other major operations that result in Quality Center-QuickTest communication. **High.** The log includes all available information related to Quality Center-QuickTest communications.
Log folder	The folder path for storing the log file. Required if a log type is specified in the **Level** option.
Restart testing tool after __ runs	For QuickTest tests, restarts QuickTest after Quality Center completes the specified number of test runs. When QuickTest restarts, it continues with the next test in the test set. You may want to use this option to maximize available memory. If you do not want QuickTest to restart during a test set run, enter 0 (default).

Option	Description
Save the open, modified test before the test run	If an existing (named) test or is open in QuickTest when the Remote Agent begins running a test, this option instructs QuickTest to save any unsaved changes to the open test or. **Note:** If an existing (named) function library is open in QuickTest when the Remote Agent begins running a test, the function library is not saved.
Save the open, new test before the test run	If a new (untitled) test is open in QuickTest when the Remote Agent begins running a test, the test is saved in: **<QuickTest installation folder>\Tests\Quality Center** with a sequential test name.
Open a new test after the test run	By default, the last test run by the Remote Agent stays open in QuickTest when it finishes running all tests. However, if any shared resources (such as a shared object repository or Data Table file) are associated with the open test, those resources are locked to other users until the test is closed. You can select this option to ensure that the last test that Quality Center runs is closed, and a blank test is open instead.

Option	Description
Run QuickTest in hidden mode	Specifies whether to run QuickTest in hidden (silent) mode when you run a test set from the Test Lab module in Quality Center. By default, this option is selected. **Display hidden-mode notification tooltip:** If this check box is selected, the Remote Agent displays a tooltip window when QuickTest runs a Quality Center test in hidden mode. You can click the tooltip to display QuickTest during the test set run. By default, this option is selected. **Notes:** ➤ Clicking the notification tooltip clears the **Run QuickTest in hidden mode** check box and QuickTest will run in normal mode. You can run QuickTest in hidden mode again by reselecting **Run QuickTest in hidden mode** before the next test set run. ➤ When running in hidden mode, QuickTest can be optionally redisplayed at the end of each test or at the end of the test set. This behavior is configured in Quality Center Site Administration using the SUPPORT_TESTSET_END parameter. For more information, see the section on setting Quality Center configuration parameters in the *HP Quality Center Administrator Guide*.

Option	Description
Restart testing tool after	Restarts QuickTest if there is no response after the specified number of seconds for: **Operations.** QuickTest operations such as Open or Run. **Queries.** Standard status queries that remote applications perform to confirm that the application is responding (such as the Quality Center **get_status** query). The default value for both options is 2700 seconds (45 minutes). However, while QuickTest operations may take a long time between responses, queries usually take only several seconds. Therefore, you may want to set different values for each of these options. **Note:** If a function library with unsaved changes is open in QuickTest, QuickTest prompts you to save it. If the function library is not saved within 10 seconds, QuickTest restarts and any unsaved changes are lost.

52

Using the Resources and Dependencies Model

QuickTest enables you to use the Resources and Dependencies model to fully integrate your QuickTest tests into Quality Center projects.

Note: The references to Quality Center features and options in this chapter apply to Quality Center 10.00. However, they may not be supported in the Quality Center edition you are using. For information on Quality Center editions, see the *HP Quality Center User Guide*.

This chapter includes:

Resources and Dependencies Model Terminology

Term	Description
Assets	Any QuickTest testing document or resource file, including: ➤ tests ➤ actions ➤ function libraries ➤ shared object repositories ➤ recovery scenarios ➤ data table files ➤ environment variable files **Note:** In Quality Center, QuickTest assets are referred to by the more general term **entities**.
Resources	Any asset used by a test. For example, a test may contain calls to functions in associated function libraries, and it may reference test objects stored in shared object repositories that are associated with the test. Resources include: ➤ function libraries ➤ shared object repositories ➤ recovery scenarios ➤ data table files ➤ environment variable files
Dependencies	The linked relationships between resources or external actions and a particular test. Associated resource files and actions are linked to each test that uses these resources or calls these actions. Assets are considered dependencies if they are associated using absolute paths, and they are stored in the following modules: ➤ **Test Plan module:** tests ➤ **Test Resources module:** function libraries, shared object repositories, recovery scenarios, data table files, environment variable files **Note:** Tests stored in the **Unattached** folder in the Test Plan module are not considered dependencies because they are not associated with any test.

About the Resources and Dependencies Model

The Resources and Dependencies model provides powerful integration between QuickTest and Quality Center.

- Replaces the use of attachments with linked QuickTest assets. You store your tests in the Test Plan module, and you store your resource files in the Test Resources module. When you associate a resource file to a test, these assets become linked. Linking assets improves runtime performance by decreasing download time. It also helps to ensure that the relationships between dependent assets are maintained (using attachments increases download time from Quality Center 10.00).
- Supports versioning and baselines for tests and resource files. You can create versions of these assets in QuickTest or in Quality Center. You manage asset versions and baselines in Quality Center. For more information, see "Managing Assets Using Version Control" on page 1479.
- Enables you to view and compare your QuickTest assets in both Quality Center and QuickTest. You can use the Asset Comparison Tool to compare versions of individual QuickTest assets and the Asset Viewer for viewing an earlier version of a QuickTest asset. Both of these viewers are available in Quality Center and in QuickTest. For more information, see "Viewing and Comparing Versions of QuickTest Assets" on page 1461.
- Enables you to import and share assets across Quality Center projects. For more information, see the *HP Quality Center User Guide*.

For more information about the advantages of working with this model, see "Advantages of Working with Asset Dependencies" on page 1451.

Considerations for Working with Relative Paths in Quality Center

Resource files and actions that are associated with tests using a relative path are not considered dependencies. To ensure that your resource files are recognized as dependencies, they must be saved in the Test Resources module in Quality Center, and they must be associated using the full Quality Center path. This enables you to benefit from the features provided by the Resources and Dependencies Model, as described in "Advantages of Working with Asset Dependencies" on page 1451.

Despite this, there may be cases in which you may want to use a relative path. For example, if your application is released in different languages, you may want to use a relative path when associating shared object repositories with your tests, as this enables you to use the same test with localized shared object repositories. Similarly, you may want to use the same tests to test different versions of your application using version-specific shared object repositories.

When assets are associated via relative paths, consider the following:

- Run-time performance times are slower.
- Dependency information for these assets is not displayed in:
  - The Using and Used By grids in the Dependencies tab in Quality Center.
    See: "The Dependencies Tab" on page 1454
  - The Used By tab of the Action Properties dialog box in QuickTest.
    See: "Viewing a List of the Tests and Actions Using this Action" on page 452
  - The message box that opens when you delete an asset that is associated with other assets.
    See: "Advantages of Working with Asset Dependencies" on page 1451
- Quality Center does not verify that these assets are included during the baseline verification process.
  See: "Viewing Baseline History" on page 1490

- When opening or running tests from a baseline, any associated external action or resource file that is associated via a relative path but is **not** included in the baseline is considered a missing resource. This may cause a test run may to fail. (Note that the baseline version of an associated asset is used if the asset associated via a relative path **is** included in the baseline.)
See: "Viewing Baseline History" on page 1490
- When using the Asset Comparison Tool to view a test, you cannot drill down to view assets that are associated via a relative path.
See: "Working with the Asset Comparison Tool and Asset Viewer" on page 1462

Advantages of Working with Asset Dependencies

When you associate a **dependent** resource file with a test, the assets become integrally linked, and these links can be viewed in Quality Center (in the Dependencies tab of various modules) and in QuickTest (in the Action Properties dialog box).

- **Assets stay linked.** When you move a test, or rename a test or action, dependent assets are automatically updated to reflect the change. This helps ensure that there are no missing resources during a run session.
- **Resource files are all stored in one Quality Center module.** Resource files are stored in the Test Resources module, enabling you to manage your resources in one central location, and to view at a glance which tests are using each resource file. For more information on the Test Resources module, see the *HP Quality Center User Guide*.
- **Using resources stored in the Test Resources module improves runtime performance.** Tests open and run faster when the associated resource files are stored in the Test Resources module (instead of being stored as attachments to tests in the Test Plan module).
- **Version control can also be applied to resource files.** When version control is enabled for a project, all of the assets can be checked into the version control database, and baselines can be created that capture the developmental stage for each asset. For more information, see "Managing Assets Using Version Control" on page 1479.

- **You can share assets with other projects and synchronize them as needed.** You can copy assets from other projects. This enables you to reuse your existing assets instead of creating new assets whenever you create a new project. For example, you can create a "template" set of assets to use as a basis for new projects.

  You can synchronize these assets in both projects when changes are made, or you can customize your assets to suit the unique needs of each development project. For more information, see the sections on importing and synchronizing libraries in the *HP Quality Center User Guide*.

- **Deleting assets is easier.** When you delete an asset (a reusable action or associated resource file), a warning message informs you if the asset is used by other tests (or more than once in the current test). This message contains a **Details** section that lists the tests that are associated with this asset or contain calls to this action so you can modify the tests as needed. This helps you manage your tests and action calls so that tests do not inadvertently fail.

- **You can verify which tests contain calls to an action.** You can view a list of the tests that contain calls to a particular action by focusing on the action and opening the Used By tab in the Action Properties dialog box. (Right-click an action in the Test Flow pane and select **Action Properties**.) For more information, see “Viewing a List of the Tests and Actions Using this Action” on page 452.

Working With the Resources and Dependencies Model in Quality Center

When you create a Quality Center project in your Quality Center server, the QuickTest tests that you create in this project are saved to the Test Plan module. The resource files associated with your tests are saved to the Test Resources module as linked dependencies.

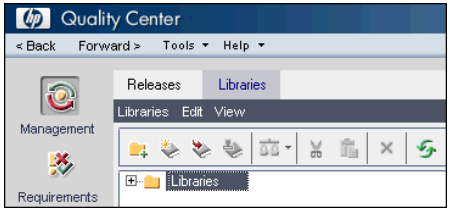
This section provides a general overview of the tabs that are relevant for QuickTest tests. For information on using any of these tabs, see the relevant section in the *HP Quality Center User Guide*.

The Libraries Tab

In the Libraries tab, you can:

- **Create, view, and compare baselines.** For more information, see "Viewing Baseline History" on page 1490 and the sections describing baselines in the *HP Quality Center User Guide*.
- **Import assets from other Quality Center projects.** This enables you to create a complete copy of the assets that are included in a baseline in another project in any accessible domain. For more information, see the *HP Quality Center User Guide*.

The Libraries tab is located to the right of the Releases tab in the Management module.



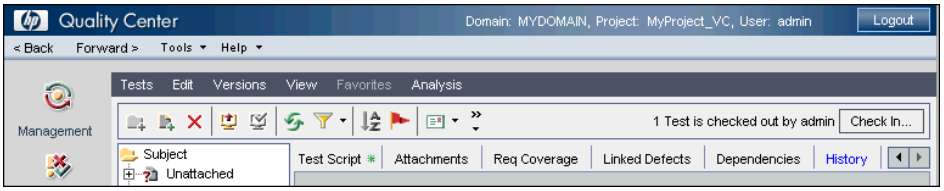
The History Tab

The History tab lists version and baseline information for a selected file. You can view and compare file versions, and you can see the baseline in which a version is stored (if applicable). You can also check out an earlier version if you want to roll back to that version. When you check the file back into the version control database, that version becomes the current version.

The History tab is available from the following modules:

- Test Plan module
- Test Resources module

The History tab is located in the pane on the right side of the window. You may need to scroll to the right to display it.



For more information on the History tab, see the *HP Quality Center User Guide*.

Tip: You can also view version history and baseline history in QuickTest by selecting **File** > **Quality Center Version Control** > **Version History** or **File** > **Quality Center Version Control** > **Baseline History**. For more information, see "Viewing Version History for an Asset" on page 1488 and "Viewing Baseline History" on page 1490.

The Dependencies Tab

The Dependencies tab displays the relationship between a selected asset, such as a test, and the assets with which it is associated. You use the Dependencies tab to see at a glance which resources are used by a particular asset, and which asset is using a particular resource.

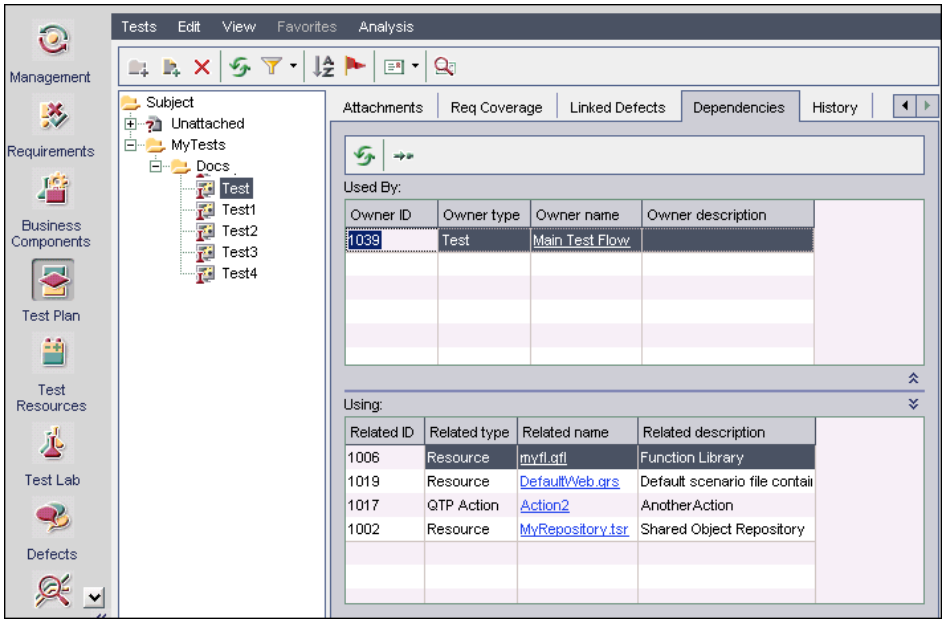
For example, suppose you want to modify the objects in a shared object repository. You can navigate to the shared object repository in the Test Resources module to view a list of the tests with which it is associated. This enables you to determine which assets this resource file is used by and helps you to analyze the impact that a proposed change may make to the dependent assets.

In Quality Center, you can view this **Using** and **Used By** information in the Dependencies tab in the Test Plan and Test Resources modules.

The Dependencies tab is available from the following modules:

- Test Plan module
- Test Resources module

Below is an example of a Dependencies tab in Quality Center:



The Dependencies tab contains the **Used By** grid and the **Using** grid. The Used By grid displays assets that depend on a selected asset. The Using grid displays the assets that a selected asset depends on.

Used By Grid

The Used By grid lists the assets that depend on the selected asset because they are using that asset. For example, suppose you are looking at the Used By grid for a shared object repository. The Used By grid lists all of the tests that are associated with this dependency.

The Used By grid contains the following columns:

Column	Description
Owner ID	A unique numeric ID assigned automatically by Quality Center. If the **Owner ID** is a link, you can click it to jump to that asset in Quality Center. **Example:** Suppose you are looking at the Used By grid for a specific function library in the Test Resources module. You can click the **Owner ID** link to jump to the test with which it is associated. (The link takes you to the Test Plan module.)
Owner Type	The type of asset that is using the selected asset. QuickTest-related owner types include: ➤ **Resource.** A resource listed in the Test Resources module. ➤ **Test.** A QuickTest test in the Test Plan module. ➤ **QTP Action.** An action that is part of a test in the Test Plan module.

Column	Description
Owner Name	The name of the asset that is using the selected asset. If the **Owner Name** is a link, you can click it to jump to this asset Quality Center. QuickTest-related owner names include: ➤ **Main Test Flow.** Indicates the test container called by the top-level action in the currently selected test in the Test Plan module. When **Main Test Flow** is displayed, the **Owner Type** is **Test** ➤ **Action<#>.** Indicates the internal name of the action that is called by an action in the currently selected test in the Test Plan module. **Action<#>** refers to the sequential number of the action when it was created. **Action<#>** is displayed when the **Owner Type** is **QTP Action**. **Note: Action<#>** is displayed even if an action was renamed in the test. ➤ The actual name of the asset if the asset is not a test.
Owner Description	The description of the associated asset that uses the selected asset. ➤ If the **Owner Type** is **QTP Action**, displays the name of the action as shown in QuickTest and displays its description, if any. ➤ If the **Owner Type** is **Test**, this cell is empty.

Tip: In QuickTest, you can view **Used By** information for actions in the **Action Properties** dialog box. For more information, see "Viewing a List of the Tests and Actions Using this Action" on page 452.

QuickTest also displays **Used By** information when you try to delete a dependent asset, so that you can determine how the change might affect associated assets before you delete the asset.

Using Grid

The Using grid lists all of the dependencies that the selected asset is using. For example, suppose you are looking at a test. You can see all of the external actions called by the test, all of the shared object repositories containing test objects used by the test, function libraries containing functions called by the test, and so on.

The Using grid contains the following columns:

Column	Description
Related ID	A unique numeric ID assigned automatically by Quality Center.
Related Type	The type of associated asset that the selected asset uses. QuickTest-related types include: ➤ **Resource.** A resource listed in the Test Resources module. ➤ **Test.** A QuickTest test in the Test Plan module. ➤ **QTP Action.** An action that is part of a test in the Test Plan module.

Column	Description
Related Name	The name of the associated asset that the selected asset uses. QuickTest-related names include: ➤ **Action<#>.** Indicates the internal name of the action that is called by an action in a test in the Test Plan module. **Action<#>** refers to the sequential number of the action when it was created. **Action<#>** is displayed when the **Related Type** is **QTP Action**. **Note: Action<#>** is displayed even if an action was renamed in the test. ➤ The actual name of the asset if the asset is not a test.
Related Description	The description of the associated asset that the selected asset uses. If the **Related Type** is **QTP Action**, displays the name of the action as shown in QuickTest and displays its description, if any.

53

Viewing and Comparing Versions of QuickTest Assets

This chapter describes how to use the Asset Comparison Tool to compare versions of QuickTest assets that are stored in Quality Center. An **asset** can be a QuickTest testing document or any resource file that is used by a QuickTest testing document.

Note: The references to Quality Center features and options in this chapter apply to Quality Center 10.00. However, they may not be supported in the Quality Center edition you are using. For information on Quality Center editions, see the *HP Quality Center User Guide*.

This chapter includes:

Tip: To compare two different object repositories, use the Object Repository Comparison Tool, described on page 287.

Working with the Asset Comparison Tool and Asset Viewer

This section describes the tasks most often performed using the Asset Comparison Tool and the Asset Viewer.

View a comparison of two asset versions (Asset Comparison Tool)

You can view comparison of two versions of an asset either side-by-side, or one above the other.

Drill down to compare or view a specific element

Compare versions of an integral element. You can view a drilldown comparison of a specific element in the currently open version comparison. Elements include any resource that is an integral part of the test (not saved as an external resource), such as the Data Table or local object repository. When you check in a test, these elements are checked in, too. This enables you to view a version comparison of these elements directly from the test.

View the latest content of an associated resource file. An associated resource file is any resource file used by an asset. For example, a function library and a shared object repository are examples of resource files that can be used by a test. When you drill down in a test, you can view the last saved version of a resource file. This enables you to view the latest content. (If you want to compare different versions of the drilled-down resource, you can open the resource and perform a new comparison.)

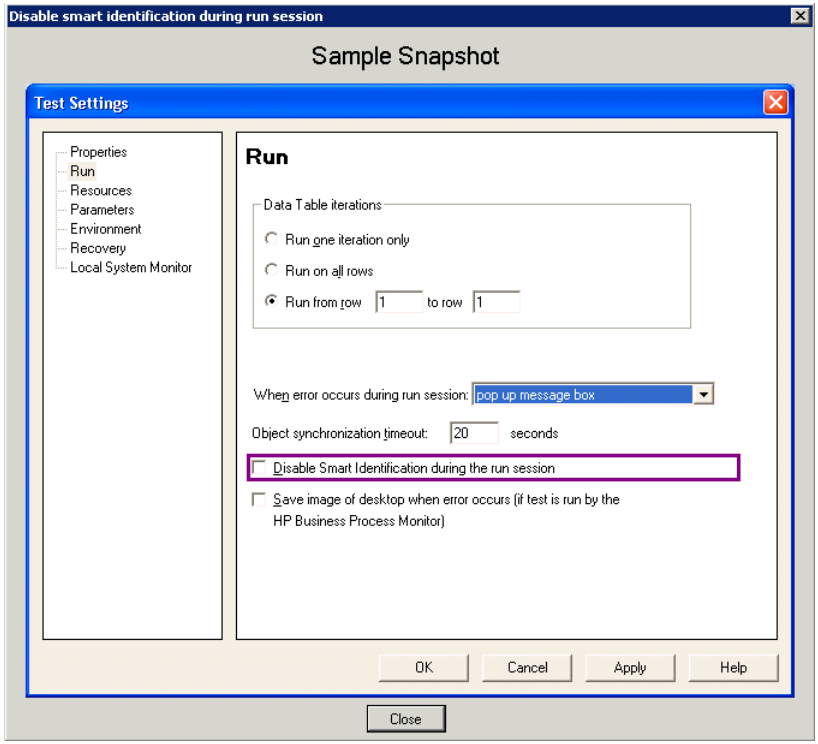
To drill down, do one of the following:

- Click the blue drilldown arrow adjacent to any asset that can be compared. (The pointer changes into a pointing hand in the proximity of the drilldown arrow.)
- Double-click the element.
- Right-click the element and select **View Drilldown of Selected Asset**. For more information, see "QuickTest Asset Comparison Tool - Context Menu Commands" on page 1472.

View the QuickTest location of an element

You can view a screen capture depicting the QuickTest location of an element by right-clicking the relevant node and selecting **View Sample Snapshot**. The screen capture displays an example of the relevant dialog box. The option (or area) for the node you right-clicked is highlighted in the screen capture.

For example, suppose you are viewing a comparison of a test, and you notice that the Disable Smart Identification during the run session node is highlighted, indicating that it was changed. If you are not sure where this option is located in QuickTest, you can right-click the node in the comparison tree and select **View Sample Snapshot**. QuickTest then opens a dialog box showing you that this area is located in the Run pane of the Test Settings dialog box. The title bar of the dialog box lists the selected element, and a purple rectangle outlines the option.



For more information, see "QuickTest Asset Comparison Tool - Context Menu Commands" on page 1472.

Modify text and background colors

You can modify the text and background colors for the filter types (changed, added, removed, and so on) in the Asset Comparison Tool window using the Color Settings dialog box.

When you modify the background color of a filter type, the color of the filter type in the legend at the top of the window changes accordingly. These changes remain in effect unless you change them again or restore the default settings.

For more information, see "The Color Settings Dialog Box" on page 1473.

View the number of differences for a specific element

If the sub-elements of an element are different between versions, and you collapse the node representing that element, a legend is displayed adjacent to the node. This legend indicates the number of differences that exist under the collapsed element. In the following example, three sub-elements were changed, one was removed, and seven were added:

For more information, see "The QuickTest Asset Comparison Tool" on page 1465 and "The QuickTest Asset Viewer" on page 1474.

The QuickTest Asset Comparison Tool

The QuickTest Asset Comparison Tool enables you to compare two versions of a particular QuickTest asset, such as a test, a function library, a shared object repository, or a recovery scenario. It also enables you to drill down in an asset to view a comparison of entities that are associated with the asset, for example, an associated Data Table or shared object repository.

Note: The QuickTest Asset Comparison Tool does not enable you to drill down to view assets that are associated via a relative path. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.

Opening the QuickTest Asset Comparison Tool

You can open the QuickTest Asset Comparison Tool from QuickTest or from Quality Center when version control is enabled, or from a command line interpreter.

You can open the Asset Comparison Tool from:

The main QuickTest window:

1. Open the test or function library whose versions you want to compare.
2. Select **File** > **Quality Center Version Control** > **Version History** or **Baseline History**. The Version History or Baseline History dialog box opens.
3. Select two versions and click **Compare**. The Asset Comparison Tool opens.

The Object Repository Manager

1. Open the Object Repository Manager (**Resources** > **Object Repository Manager**).
2. Browse to and open the shared object repository whose versions you want to compare. For more information, see "Opening Object Repositories" on page 217.

3 Select **File** > **Quality Center Version Control** > **Version History** or **Baseline History**. The Version History or Baseline History dialog box opens.

4 Select two versions and click **Compare**. The Asset Comparison Tool opens.

The Recovery Scenario Manager:

1 Open the Recovery Scenario Manager (**Resources** > **Recovery Scenario Manager**).

2 Open the recovery scenario file whose versions you want to compare. For more information, see "Understanding the Recovery Scenario Manager Dialog Box" on page 1336.

3 Click the **Version Control** down arrow and select **Version History** or **Baseline History**.

4 Select two versions and click **Compare**. The Asset Comparison Tool opens.

Quality Center:

1 In Quality Center, connect to the project containing the asset you want to compare.

2 Do one of the following:

- Click the **Test Plan** button in the sidebar to open the Test Plan module.
- Click the **Test Resources** button in the sidebar to open the **Test Resources** module. This module contains the resource files associated with your test, such as function libraries, shared object repositories, Data Tables, and recovery scenarios.

3 In the tree, select the file whose versions you want to compare.

4 Click the **History** tab, and then click the **Versions and Baselines** tab.

5 In the **View by** box, select either **Versions** or **Baselines**.

6 In the grid, select two versions to compare, and then click the **Compare** button.

7 In the sidebar of the window that opens, click the **QTP Comparison** button. The Asset Comparison Tool opens.

Tip: You can also compare baselines from the **Management** module. Click the **Management** button in the side bar to open the **Management** module. Select a baseline in the tree and click the **Compare To** button. For more information, see the *HP Quality Center User Guide*. For more information on baselines, see "Managing Versions of Assets in Quality Center" on page 1480.

The Command Line Interpreter (cmd.exe):

1 Open the Command Line Interpreter.

2 Enter the command using the following syntax:

"<**Asset Comparison Tool executable path**>" **P1:** "<**file path 1**>" **P2:** "<**file path 2**>"

where **P1** = the file system path to the first asset, and **P2** = the file system path to the second asset.

Note: Make sure you insert a blank space after each argument. The options are not case-sensitive and can be entered in any order.

Example:

```
"C:\Program Files\HP\QuickTest Professional\Bin\QTPDiffApplication.exe" P1: "C:\Program Files\HP\
QuickTest Professional\Tests\Test1" P2: "C:\Program Files\HP\QuickTest Professional\Tests\Test2"
```

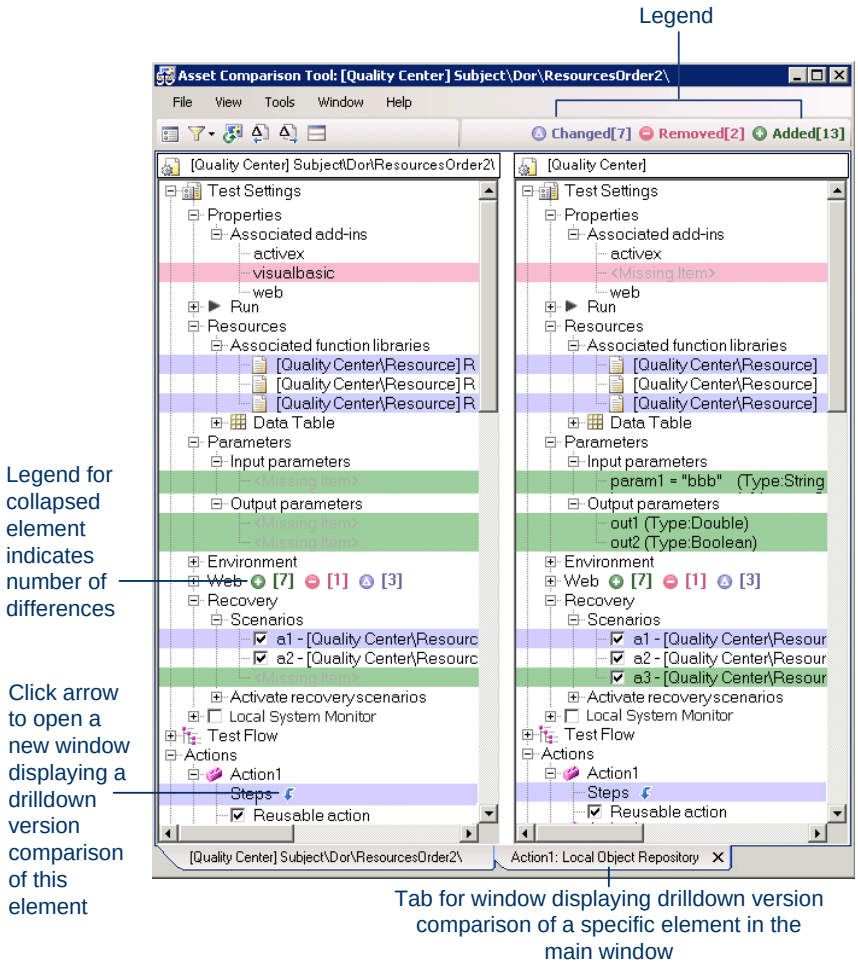
Understanding the Asset Comparison Tool Commands and Options

This section describes the commands and options available in the Asset Comparison Tool and the Asset Viewer. For an overview of these tools, see "Working with the Asset Comparison Tool and Asset Viewer" on page 1462.

The Asset Comparison Tool enables you to compare two versions of an asset. For more information, see "The QuickTest Asset Comparison Tool" on page 1465.

The Asset Viewer enables you to view a particular of an asset. For more information, see “The QuickTest Asset Viewer” on page 1474.

Below is an image of the QuickTest Asset Comparison Tool.



QuickTest Asset Comparison Tool - Menu, Toolbar, and Button Options

	Commands	Shortcut Key	Description
	File > Exit	ALT+F4	Closes the Asset Comparison Tool window.
	**View > Next Difference**	CTRL+ DOWN ARROW	Finds the next difference between the elements in the compared versions.
	**View > Previous Difference**	CTRL+UP ARROW	Finds the previous difference between the elements in the compared versions.
	**View > Refresh**		Performs a new comparison of the selected asset versions. **Note:** This is useful if you are comparing the current version of an asset. If you modify and save the asset, you can use the **Refresh** command to view an updated comparison.
	**Tools > Color Settings**		Opens the Color Settings dialog box, enabling you to define the text and background color for each filter type. See: "The Color Settings Dialog Box" on page 1473

	Commands	Shortcut Key	Description
	**Tool > Filter**		Enables you to show or hide the following types of filter elements in the comparison window: ➤ **Changed** ➤ **Removed** ➤ **Added** ➤ **Identical** Select or clear a filter command. The comparison window displays only those elements that match the defined filter. **Tip:** The legend in the top-right corner of the window indicates how many elements match each filter type. The legend adjacent to a collapsed node indicates how many sub-nodes match each filter type.
	Window > Close Window		Closes the currently active comparison window if it was opened from the main comparison window. Enabled only if more than one comparison window is open. **Note:** You can open another window to view a comparison of an asset that is associated with the currently compared asset, such as a shared object repository or Data Table. You do this by clicking the blue drilldown arrow adjacent to any asset that can be compared. **Tip:** You can also close the comparison window by clicking the **X** in the tab at the bottom of the window.
	**Window > View Horizontal or View Vertical**		**View Horizontal.** Displays the open documents one above the other. **View Vertical.** Displays the open documents side-by-side.

	Commands	Shortcut Key	Description
	Window > <Compared Asset Path>		Enables you to navigate between the open comparison windows.
	Help > Asset Comparison Tool Help	F1	Opens the Asset Comparison Tool Help.
	Previous 2000 Lines button		If the testing document has more than 2000 lines, this button is displayed at the top of the comparison pane. Click to hide the current 2000 lines and display the previous 2000 lines of the testing document.
	Next 2000 Lines button		If the testing document has more than 2000 lines, this button is displayed at the bottom of the comparison pane. Click to hide the current 2000 lines and display the next 2000 lines of the testing document.

QuickTest Asset Comparison Tool - Legend

The following is an example of the filter legend displayed in the top-right corner of the Asset Comparison Tool window:

Symbol	Description	Number
	Changed	Indicates the number of modified elements in the comparison.
	Removed	Indicates the total number of elements that were removed from either of the versions being compared.
	Added	Indicates the total number of elements that were added to either of the versions being compared.

Notes:

- If you modify the background color of a filter type (using the Color Settings dialog box), the color of the filter type in the legend changes accordingly.
- If you collapse an asset in the comparison window, the tool displays a legend for that asset, as shown in the following example:

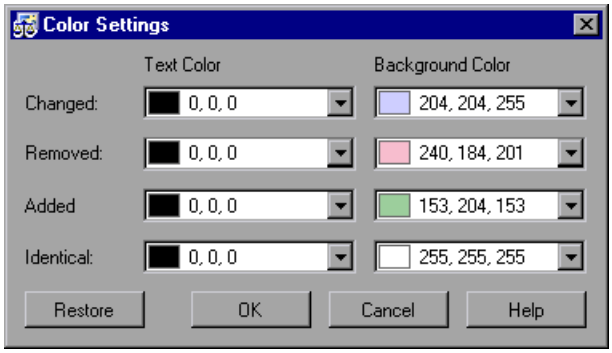
QuickTest Asset Comparison Tool - Context Menu Commands

Command	Shortcut Key	Description
View Drilldown of Selected Asset	ENTER	Opens a drilldown version comparison of the selected asset in a new window. (Relevant only for assets that can be compared.) **Tip:** You can also click the blue drilldown arrow adjacent to the node to open a drilldown version comparison in a new window. **Note:** You cannot drill down to view assets that are associated via a relative path. See: "Considerations for Working with Relative Paths in Quality Center" on page 1450
View Sample Snapshot	CTRL+Q	Opens a window containing a sample image of the selected element in QuickTest, for example, the Resources pane in the Test Settings dialog box. The element itself is highlighted in the snapshot.

The Color Settings Dialog Box

Description	Enables you to modify the text and background colors for the various filter elements in the Asset Comparison Tool window. The changes remain in effect for all subsequent sessions. **Note:** If you change the background color for a filter type, the legend in the top-right corner of the Asset Comparison Tool window changes accordingly.
How to Access	In the Asset Comparison Tool window: ➤ Select the **Tools** > **Color Settings** menu command. ➤ Click the **Color Settings** toolbar button .

Below is an image of the Color Settings dialog box:



Color Settings Dialog Box Options

Option	Description
Added **Removed** **Changed** **Identical**	Choose a text color and background color for the relevant filter elements. You can: ➤ Click a down arrow to select a color from the list of colors in the from the Custom, Web, or System tabs. ➤ Enter an RGB value directly in the edit box.
Restore	Click to restore the default color values for each of the filter elements.

The QuickTest Asset Viewer

The QuickTest Asset Viewer enables you to view an earlier version of a particular QuickTest asset, such as a test, a function library, a shared object repository, or a recovery scenario. You can also drill down in the Asset Viewer window to view associated entities, such as an associated Data Table or shared object repository.

Opening the QuickTest Asset Viewer

You can open the QuickTest Asset Viewer from QuickTest or from Quality Center when version control is enabled.

You can open the Asset Viewer from:

The main QuickTest window:

1. Open the test or function library for which you want to view an earlier version.
2. Select **File** > **Quality Center Version Control** > **Version History**. The Version History dialog box opens.
3. Select a version and click **View**. The Asset Viewer opens.

The Object Repository Manager:

1. Open the Object Repository Manager (**Resources** > **Object Repository Manager**).
2. Browse to and open the shared object repository for which you want to view an earlier version. For more information, see "Opening Object Repositories" on page 217.
3. Select **File** > **Quality Center Version Control** > **Version History**. The Version History dialog box opens.
4. Select a version and click **View**. The Asset Viewer opens.

The Recovery Scenario Manager:

1 Open the Recovery Scenario Manager (**Resources** > **Recovery Scenario Manager**).

2 Open the recovery scenario file for which you want to view an earlier version. For more information, see "Understanding the Recovery Scenario Manager Dialog Box" on page 1336.

3 Click the **Version Control** down arrow and select **Version History**.

4 Select a version and click **View**. The Asset Viewer opens.

Quality Center:

1 In Quality Center, connect to the project containing the asset you want to view.

2 Do one of the following:

- Click the **Test Plan** button in the sidebar to open the Test Plan module.
- Click the **Test Resources** button to open the **Test Resources** module. This module contains the resource files associated with your test, such as function libraries, shared object repositories, Data Tables, and recovery scenarios.

3 In the tree, select the file for which you want to view an earlier version.

4 Click the **History** tab, and then click the **Versions and Baselines** tab.

5 In the **View by** box, select **Versions**.

6 In the grid, select a version, and then click the **View** button. (You cannot view a version that is currently checked out.) A window opens with buttons in the sidebar enabling you to access version-specific information for the selected asset. (These buttons are identical to the tabs displayed in the right pane of the main window for the latest version of the selected asset.) For more information, see the *HP Quality Center User Guide*.

The Command Line Interpreter (cmd.exe):

Note: You use the Asset Comparison Tool executable path to open the Asset Viewer.

---

1. Open the Command Line Interpreter.
2. Enter the command using the following syntax:

   "<**Asset Comparison Tool executable path**>" **P1**: "<**file path 1**>"

   where **P1** = the file system path to the first asset.

Note: Make sure you insert a blank space after each argument. The options are not case-sensitive and can be entered in any order.

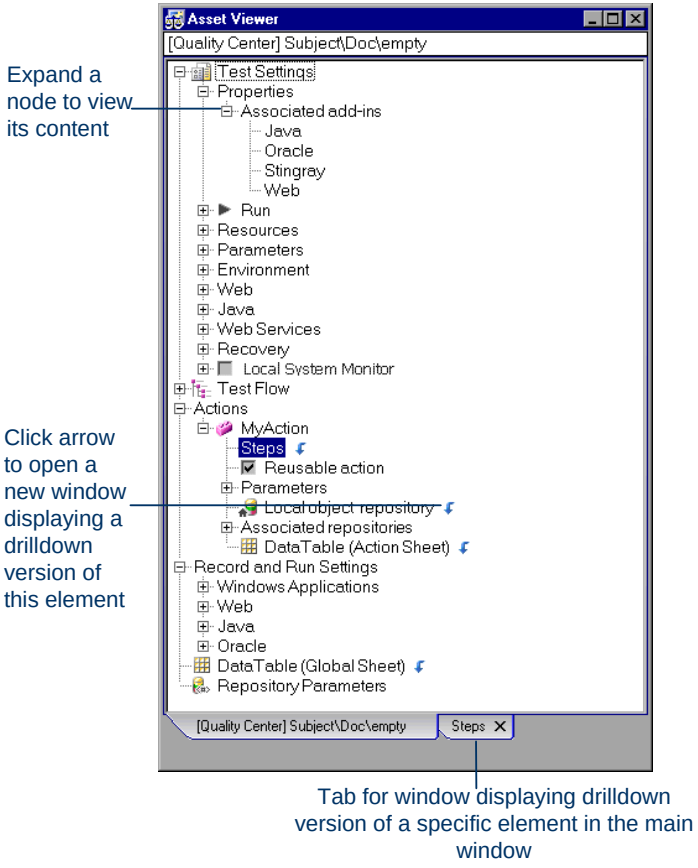
Example:

```
"C:\Program Files\HP\QuickTest Professional\Bin\QTPDiffApplication.exe" P1: "C:\Program Files\HP\QuickTest Professional\Tests\Test1"
```

Using the QuickTest Asset Viewer

The Asset Viewer provides a functional overview of an asset, enabling to view its configurations and settings in a viewer format. The tree view enables you to drill down to view or verify a particular setting without needing to open different dialog boxes or even QuickTest.

Below is an image of the QuickTest Asset Viewer:



QuickTest Asset Viewer - Button Options

Commands	Shortcut Key	Description
Previous 2000 Lines		If the testing document has more than 2000 lines, this button is displayed at the top of the pane. Click to hide the current 2000 lines and display the previous 2000 lines of the testing document.
Next 2000 Lines		If the testing document has more than 2000 lines, this button is displayed at the bottom of the pane. Click to hide the current 2000 lines and display the next 2000 lines of the testing document.

QuickTest Asset Viewer - Context Menu Commands

Command	Shortcut Key	Description
View Drilldown of Selected Asset	ENTER	Opens a drilldown version comparison of the selected asset in a new window. (Relevant only for assets that can be compared.) **Tip:** You can also click the blue drilldown arrow adjacent to the node to open a drilldown version comparison in a new window. **Note:** You cannot drill down to view assets that are associated via a relative path. See: "Considerations for Working with Relative Paths in Quality Center" on page 1450
View Sample Snapshot	CTRL+Q	Opens a window containing a sample image of the selected element in QuickTest, for example, the Resources pane in the Test Settings dialog box. The element itself is highlighted in the snapshot.

54

Managing Assets Using Version Control

This chapter describes how to use version control to manage and work with your QuickTest assets that are stored in Quality Center.

Note: The references to Quality Center features and options in this chapter apply to Quality Center 10.00. However, they may not be supported in the Quality Center edition you are using. For information on Quality Center editions, see the *HP Quality Center User Guide*.

This chapter includes:

Managing Versions of Assets in Quality Center

When QuickTest is connected to a Quality Center project with version control support, you can update and revise your QuickTest assets while maintaining earlier versions of each asset. This helps you keep track of the changes made to each asset and see what was modified from one version to another. Assets can include tests, function libraries, shared object repositories, recovery scenarios, and external Data Tables.

You manage asset versions by checking assets in and out of the version control database. You add an asset to the version control database by saving it in a Quality Center project with version control support. When you save an asset for the first time, QuickTest automatically checks the asset into the Quality Center version control database, assigns it version number 1, and automatically checks the asset out for you so that you can continue working on it. When you check the asset in, the asset retains version number 1, since this is the first version that can contain content. Then, each time the asset is checked out and in again, the version number increases by 1.

Note: If you create an asset directly in Quality Center, the asset is assigned version number 1 and is immediately checked out to you. In Quality Center, version number 1 represents the created asset without content. When you next check the asset in, Quality Center assigns it version number 2.

You can check in the asset at any time. For example, you may want to check the asset in every day or when you complete a task. While the asset is checked out to you, other users can view the last checked in version of that asset in read-only mode, but they cannot modify the asset or view your changes until you check in the asset.

If the asset is...	You can...
checked in	➤ Open the asset in read-only mode using the **Open** option. You cannot modify the asset. ➤ Open the asset and check it out immediately using the **Open and Check out** option. You can modify the asset as needed.
checked out to your Quality Center user name	Open the asset using the **Open** option and modify the asset as needed.
checked out to another Quality Center user	Open the asset in read-only mode using the **Open** option. QuickTest displays a message indicating that the asset is checked out to another Quality Center user. You view the last checked in version of the asset now, and you can check out the asset later after the other user checks in the asset.

In QuickTest, you can check out only the latest version of an asset, although you can view and compare earlier versions. This is because assets that are stored in Quality Center are often linked to or **dependent on** one another.

For example, if you try to run an earlier version of a test with a later version of a shared object repository, your test might fail because the objects in the object repository would not necessarily match the objects in the test.

If you need to check out an earlier version of an asset, for example, to roll back to an earlier version, contact your Quality Center project administrator. Your administrator needs to ensure that the correct versions of all relevant assets become the latest versions.

You can view and compare the versions of an asset using the Asset Comparison Tool. For more information, see "Viewing and Comparing Versions of QuickTest Assets" on page 1461.

If your project administrator creates project baseline versions when a milestone is reached during product development, you can view and compare the asset versions stored in these baselines. For more information, see "Viewing Baseline History" on page 1490.

Note: With the exception of the **Baseline History** option, the **Quality Center Version Control** options in the **File** menu are available only when you are connected to a Quality Center project with version control support, and an asset stored in Quality Center is open in the QuickTest window.

Version Management Commands

The following version control commands are available in QuickTest:

- **Check Out.** Enables you to check a version-controlled asset out of the version control database. For more information, see "Checking Assets Out of the Version Control Database" on page 1483.
- **Undo Check Out.** Enables you to cancel the check out of a version-controlled asset from the version control database. For more information, see "Canceling a Check-Out Operation" on page 1487.
- **Check In.** Enables you to check an asset in to the version control database. For more information, see "Checking Assets Out of the Version Control Database" on page 1483.
- **Version History.** Enables you to view or compare the versions of a particular asset. For more information, see "Managing Versions of Assets in Quality Center" on page 1480.
- **Baseline History.** Enables you to view or compare the versions of a particular asset as it was saved in a project's baselines. For more information, see "Viewing Baseline History" on page 1490.

Adding Assets to the Version Control Database

When you use **Save As** to save a new asset in a Quality Center project with version control support, QuickTest automatically saves the asset in the project, checks the asset into the version control database with version number 1, and then checks it out so that you can continue working. This is an administrative version of the asset, similar to a placeholder. The version number indicates that the asset exists in the database. When you later check in the asset, the version number remains version number 1—the first version that you are checking in. Subsequent checkins increase the version number by 1.

Saving your changes to an existing asset does not check them in. Even if you save and close the asset, the asset remains checked out until you choose to check it in. For more information, see "Checking Assets into the Version Control Database" on page 1486.

Checking Assets Out of the Version Control Database

When you open an asset that is currently checked in to the version control database, it is opened in read-only mode. You can review the checked-in asset. You can also run the asset and view the results.

To modify the asset, you must check it out. When you check out an asset, Quality Center copies the asset to your unique check-out directory (automatically created the first time you check out an asset), and locks the asset in the project database. This prevents other users of the Quality Center project from overwriting any changes you make to the asset. However, other users can still run the version that was last checked in to the database.

You can save and close the asset, but it remains locked until you return the asset to the Quality Center database. To release the asset, either check the asset in, or undo the check out operation. For more information on checking assets in, see "Checking Assets into the Version Control Database" on page 1486. For more information on undoing the check-out, see "Canceling a Check-Out Operation" on page 1487.

In QuickTest, the check out option accesses the latest version of the asset. In Quality Center, you can also check out earlier versions of the asset. For more information, see "The Version History Dialog Box" on page 1488 and *HP Quality Center User Guide*.

Before you check out an asset, make sure the asset you want to check out is currently checked in. If you open an asset that is checked out to you, the **Check Out** option is disabled. If you open an asset that is checked out to another user, all Quality Center version control options, except the **Version History** option, are disabled.

Note about version numbers: Prior to Quality Center 10.00, version numbers consisted of three segments separated by periods, for example 1.7.4. From Quality Center 10.00, version numbers consist of a single segment, for example 12.

To check out the latest version of an asset using the Open dialog box:

1 Do one of the following:

If the asset is a:	Do this:
Test or Function Library	In the main QuickTest window, select **File** > **Open** > **Test** or **Function Library**, or click the **Open** down arrow and select the asset type from the list.
Shared Object Repository	In the Object Repository Manager, select **File** > **Open** or click the **Open** button.
Recovery Scenario	In the Recovery Scenario Manager, click the **Open** button.

The Open <Asset type> dialog box opens.

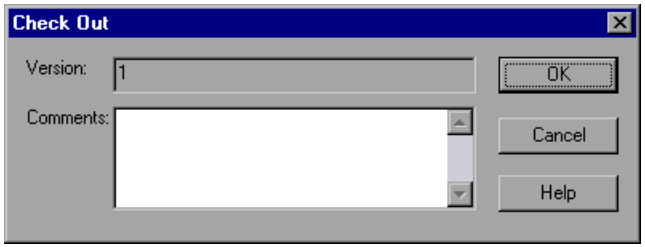
2 Browse to and select the asset.

3 Click the **Open** down arrow and select **Open and Check out**. The asset opens, checked out to you.

To check out the latest version of an asset using the File menu:

1 Open the asset you want to check out.

2 Select **File** > **Quality Center Version Control** > **Check Out**. The Check Out dialog box opens and displays the asset version to be checked out.



3 You can enter a description of the changes you plan to make in the **Comments** box.

4 Click **OK**. The read-only asset closes and automatically reopens as a writable asset.

5 View or edit your asset as necessary.

Note: You can save changes and close the asset without checking the asset in, but your changes will not be available to other Quality Center users until you check it in. If you do not want to check your changes in, you can undo the check-out. For more information on checking assets in, see "Checking Assets into the Version Control Database" on page 1486. For more information on undoing the check-out, see "Canceling a Check-Out Operation" on page 1487.

Checking Assets into the Version Control Database

While an asset is checked out, Quality Center users can run the previously checked-in version of your asset. For example, suppose you check out version 3 of an asset and make a number of changes to it and save the asset. Until you check the asset back into the version control database as version 4, Quality Center users can continue to run version 3.

When you have finished making changes to an asset and you are ready for Quality Center users to use your new version, you check it in to the version control database.

Note: If you do not want to check your changes into the Quality Center database, you can undo the check-out operation. For more information, see "Canceling a Check-Out Operation" on page 1487.

When you check an asset back into the version control database, Quality Center deletes the asset copy from your checkout directory and unlocks the asset in the database so that the asset version will be available to other users of the Quality Center project.

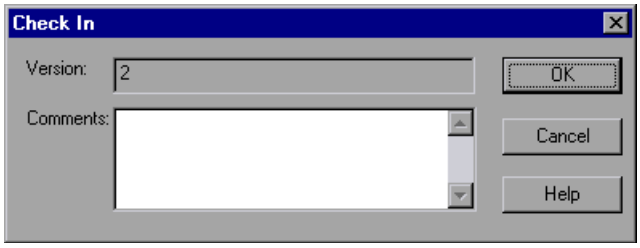
To check in the currently open asset:

1 Confirm that the currently open asset is checked out to you. For more information, see "Viewing Version History for an Asset" on page 1488.

Note: If the open asset is currently checked in, the **Check In** option is disabled. If you open an asset that is checked out to another user, all **Quality Center Version Control** options, except the **Version History** option, are disabled.

---

2 Select **File** > **Quality Center Version Control** > **Check In**. The Check In dialog box opens.



If you entered a description of your change when you checked out the asset, the description is displayed in the **Comments** box. You can enter or modify the comments in the box.

3 Click **OK** to check in the asset. The asset closes and automatically reopens as a read-only test.

Canceling a Check-Out Operation

If you check out an asset and then decide that you do not want to upload the modified asset to Quality Center, you should cancel the check out operation so that the asset will be available for check out by other Quality Center users.

To cancel a check out operation:

1 If it is not already open, open the checked out asset.

2 Select **File** > **Quality Center Version Control** > **Undo Check out**.

3 Click **Yes** to confirm the cancellation of your check out operation. The check out operation is cancelled. The checked out asset closes, and the previously checked in version reopens in read-only mode.

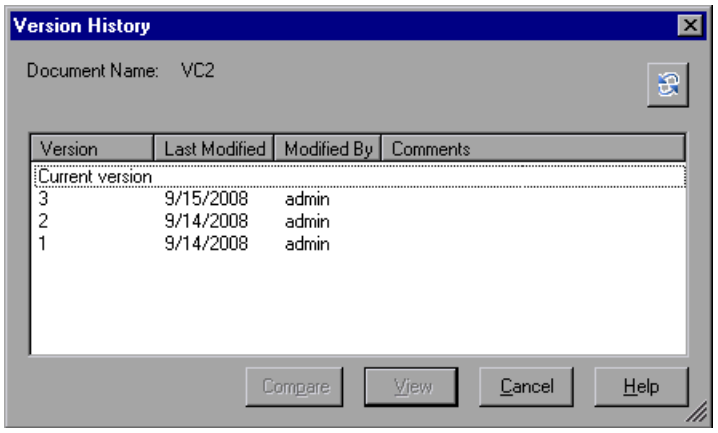
Viewing Version History for an Asset

You view the version history for an asset using the Version History dialog box. This enables you to view and compare different versions of an asset at various stages in its development.

The Version History Dialog Box

Description	Enables you to view the version history for an asset, view the content of a previous asset version, and compare two asset versions. **To view a version for an asset:** Select a version and click **View**. **To compare two versions of an asset:** Select two versions and click **Compare**.
How to Access	➤ **Most assets:** Open the asset and select the **File** > **Quality Center Version Control** > **Version History** menu command. ➤ **Recovery scenario:** In the Recovery Scenario Manager, open the recovery scenario, click the **Version Control** down arrow, and select **Version History.**
Learn More	**Conceptual overview**: "Managing Versions of Assets in Quality Center" on page 1480 **Related User Interface Topics:** "The Baseline History Dialog Box" on page 1491

Below is an image of the Version History dialog box:



Version History Dialog Box Options

	Option	Description
	Document Name	The name of the currently open asset.
	**Refresh** button	Reloads the versions in the Version History dialog box with the latest changes.
	Version column	A list of all versions of the asset.
	Last Modified column	The date that each version was checked in.
	Modified By column	The user who checked in each listed version.
	Comments column	The comments that were entered when the selected asset version was checked in.

	Option	Description
	Compare button	Enables you to compare two versions of the currently open asset. **To compare two versions:** Select the versions you want to compare and click **Compare**. QuickTest opens the two asset versions in the Asset Comparison Tool. For more information, see "The QuickTest Asset Comparison Tool" on page 1465.
	View button	Enables you to view the selected version of the current asset. **To view a version of an asset:** Select an asset version and click **View**. QuickTest opens the checked in version of the asset in the Asset Viewer. For more information on the Asset Viewer, see "The QuickTest Asset Viewer" on page 1474.

Viewing Baseline History

In Quality Center, a project administrator can create baselines that provide "snapshots" of an entire project (or part of a project) at different stages of development. A **baseline** represents a version of a project at a specific point in a project's life cycle. For example, baselines are often created for each milestone or when specific phases in a project are completed. Baselines can be created for Quality Center projects that are enabled for version control, and for projects for which version control is not enabled.

The project administrator creates the baseline in the Libraries tab of the Management module in Quality Center. Creating a baseline is a two-fold process. The administrator first creates a library, which specifies the root folders from which to import the data. The administrator makes sure to include all of the associated resource files, such as shared object repositories and function libraries. The administrator then creates the actual baseline, which comprises the latest versions of every asset included in the library. If the project is version control-enabled, then these are the latest checked in versions of every asset.

During the creation process, Quality Center verifies that all of these assets (such as associated resource files) are included in the baseline. If any assets are not included, Quality Center informs the administrator so that the library and baseline can be modified accordingly. For more information, see the *HP Quality Center User Guide*.

In Quality Center, these baselines can be viewed and compared in their entirety.

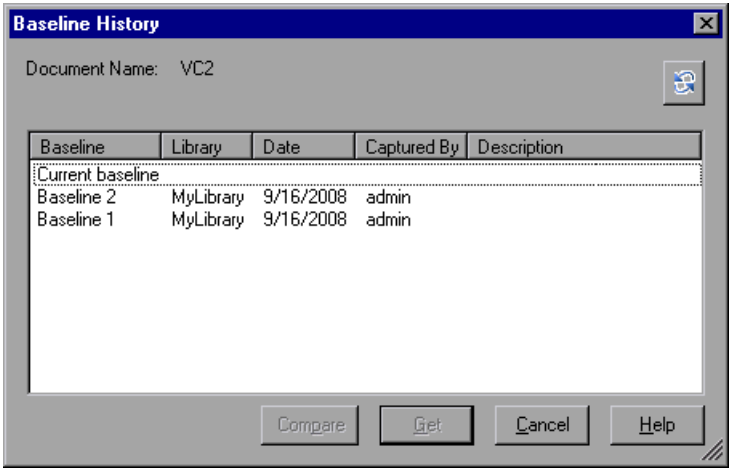
In QuickTest, you can view and compare the assets saved in these baselines. This enables you to review the content of an asset at a specific phase in the project time line.

You can also run a test from a baseline.

The Baseline History Dialog Box

Description	Enables you to view and compare read-only baseline "snapshots" of the asset.
How to Access	➤ **Most assets:** Open the asset and select the **File** > **Quality Center Version Control** > **Baseline History** menu command. ➤ **Recovery scenario:** In the Recovery Scenario Manager, open the recovery scenario, click the **Version Control** down arrow, and select **Baseline History.**
Important Information	In the Quality Center Test Lab module, you can use the **Pin to Baseline** option to run a baseline version of an asset. For more information, see the *HP Quality Center User Guide*.
Learn More	**Conceptual overview**: "Viewing Baseline History" on page 1490 **Related User Interface Topics:** "The Version History Dialog Box" on page 1488

Below is an image of the Baseline History dialog box:



Baseline History Dialog Box Options

	Option	Description
	Document Name	Specifies the name of the currently open asset.
	**Refresh** button	Reloads the baselines in the Baseline History dialog box with the latest changes. For example, if a baseline is added while this dialog box is open, clicking **Refresh** updates the list of baselines.
	Baseline column	Lists all of the baselines that include this asset. Baselines are defined in the Quality Center project (**Management** module > **Libraries** tab).
	Library column	Lists the libraries from which each baseline was created.
	Date column	Lists the date that each baseline was created.
	Captured By column	Lists the Quality Center user who created each listed baseline.
	Description column	Displays any comments that were added when the baseline was created.

	Option	Description
	Compare button	Enables you to view a comparison of the currently open asset in two baselines. **To compare two baselines:** Select the baselines you want to compare and click **Compare**. QuickTest opens the two baseline versions of the asset in the Asset Comparison Tool. For more information, see "The QuickTest Asset Comparison Tool" on page 1465.
	Get button	Enables you to open the current asset from the selected baseline. **To view the asset as it was stored in a baseline:** Select a baseline from the list and click **View**. When you click **Get**, QuickTest: ➤ Closes the currently open asset. ➤ Opens the same asset from the baseline you selected. ➤ Loads the baseline version of the external actions and resource files that are associated with the asset, if any, when they are called. **Note:** If an external action or resource file is associated via a relative path, loads the latest version of the action or resource file instead of the version from the baseline.

Version History Versus Baseline History

This section focuses on the differences between version history and baseline history and describes when to use each.

- You use version control to check in and check out assets as needed. For example, you may want to check in an asset on a daily basis or only when significant results are achieved. This enables you to monitor the asset's development.

  If you want to view the content of an asset on a particular date or after a particular user checked in the asset, use the Version History option to view or compare the asset.

- The Quality Center project administrator creates baselines that represent "snapshots" of a project's assets at various milestones in a project's life cycle. Each baseline links to the assets specified by the administrator when the baseline was created. The asset version represented in the baseline is always the version that was checked in when the baseline was created.

  If you want to view an asset as it was saved for a particular milestone, use the Baseline History option.

55

Working with Version Control in Quality Center 9.x

This chapter describes how HP Quality Center, the centralized quality solution, can help you organize and control the testing process.

Note: References to Quality Center features and options in this chapter apply to Quality Center 9.x. However, they may not be supported in the Quality Center 9.x version you are using. For more information on Quality Center editions, see the *HP Quality Center User Guide*.

This chapter includes:

Opening Tests from a Quality Center 9.x Project with Version Control Support

When you click the **Open** toolbar button or choose **File** > **Open** > **Test** to open a test from a Quality Center project with version control support, the Open Test dialog box displays icons that indicate the version control status of each test in the selected subject.

When you open a test from a Quality Center project with version control support, the test opens in read-write or read-only mode depending on the current version control status of the test:

- If the test is currently checked into the version control database or is checked out to another user, the test opens in read-only mode.
- If the test is checked out to you, the test opens in read-write mode.

Managing Test Versions in QuickTest

When QuickTest is connected to a Quality Center 9.x project with version control support, you can update and revise your automated test scripts while maintaining earlier versions of each test. This helps you keep track of the changes made to each test, see what was modified from one version of a test to another, or return to a previous version of the test.

You add a test to the version control database by saving it in a project with version control support. You manage test versions by checking tests in and out of the version control database.

The test with the latest version is the test that is located in the Quality Center test repository and is used by Quality Center for all test runs.

Notes:

- A Quality Center server with version control support requires the installation of version control software as well as the Quality Center Version Control Add-in. For more information, see your Quality Center documentation.
- The **Quality Center Version Control** options in the **File** menu are available only when you are connected to a Quality Center project database with version control support and you have a Quality Center test open.

Adding Tests to the Version Control Database

When you use **Save As** to save a new test in a Quality Center project with version control support, QuickTest automatically saves the test in the project, checks the test into the version control database with version number 1.1.1 and then checks it out so that you can continue working.

The QuickTest status bar indicates each of these operations as they occur. Note, however, that saving your changes to an existing test does not check them in. Even if you save and close the test, the test remains checked out until you choose to check it in. For more information, see "Checking Tests into the Version Control Database" on page 1499.

Checking Tests Out of the Version Control Database

When you choose **File** > **Open** > **Test** to open a test that is currently checked in to the version control database or is checked out to another user, it is opened in read-only mode.

You can review the checked-in test. You can also run the test and view the results.

To modify the test, you must check it out. When you check out a test, Quality Center copies the test to your unique check-out directory (automatically created the first time you check out a test), and locks the test in the project database. This prevents other users of the Quality Center project from overwriting any changes you make to the test. However, other users can still run the version that was last checked in to the database.

You can save and close the test, but it remains locked until you return the test to the Quality Center database. To release the test either check the test in, or undo the check out operation. For more information on checking tests in, see "Checking Tests into the Version Control Database" on page 1499. For more information on undoing the check-out, see "Canceling a Check-Out Operation" on page 1504.

By default, the check out option accesses the latest version of the test. You can also check out earlier versions of the test. For more information, see "Using the Version History Dialog Box" on page 1501.

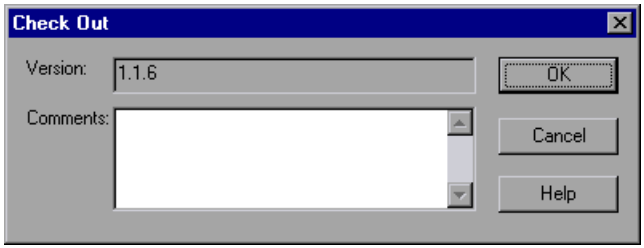
To check out the latest version of a test:

1 Open the test you want to check out. For more information, see "Opening Tests from a Quality Center 9.x Project with Version Control Support" on page 1496.

Note: Make sure the test you open is currently checked in. If you open a test that is checked out to you, the **Check Out** option is disabled. If you open a test that is checked out to another user, all **Quality Center Version Control** options, except the **Version History** option, are disabled.

---

2 Choose **File** > **Quality Center Version Control** > **Check Out**. The Check Out dialog box opens and displays the test version to be checked out.



3 You can enter a description of the changes you plan to make in the **Comments** box.

4 Click **OK**. The read-only test closes and automatically reopens as a writable test.

5 View or edit your test as necessary.

Note: You can save changes and close the test without checking the test in, but your changes will not be available to other Quality Center users until you check it in. If you do not want to check your changes in, you can undo the check-out. For more information on checking tests in, see "Checking Tests into the Version Control Database" on page 1499. For more information on undoing the check-out, see "Canceling a Check-Out Operation" on page 1504.

Checking Tests into the Version Control Database

While a test is checked out, Quality Center users can run the previously checked-in version of your test. For example, suppose you check out version 1.2.3 of a test and make a number of changes to it and save the test. Until you check the test back in to the version control database as version 1.2.4 (or another number that you assign), Quality Center users can continue to run version 1.2.3.

When you have finished making changes to a test and you are ready for Quality Center users to use your new version, you check it in to the version control database.

Note: If you do not want to check your changes into the Quality Center database, you can undo the check-out operation. For more information, see "Canceling a Check-Out Operation" on page 1504.

When you check a test back into the version control database, Quality Center deletes the test copy from your checkout directory and unlocks the test in the database so that the test version will be available to other users of the Quality Center project.

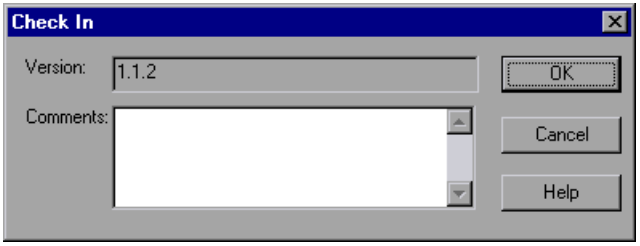
To check in the currently open test:

1 Confirm that the currently open test is checked out to you. For more information, see "Viewing Version Information For a Test" on page 1501.

Note: If the open test is currently checked in, the **Check In** option is disabled. If you open a test that is checked out to another user, all **Quality Center Version Control** options, except the **Version History** option, are disabled.

---

2 Choose **File** > **Quality Center Version Control** > **Check In**. The Check In dialog box opens.



If you entered a description of your change when you checked out the test, the description is displayed in the **Comments** box. You can enter or modify the comments in the box.

3 Click **OK** to check in the test. The test closes and automatically reopens as a read-only test.

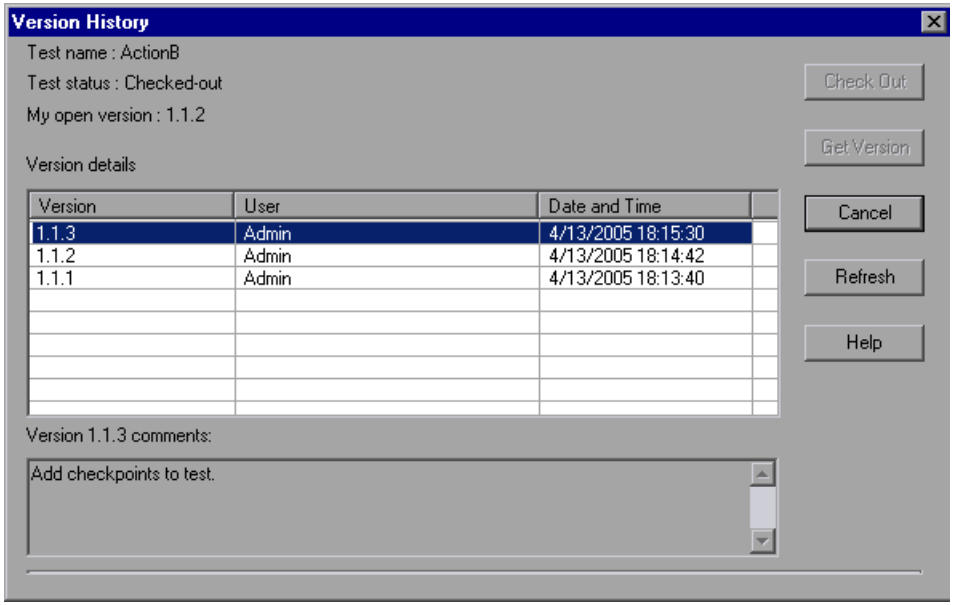
Using the Version History Dialog Box

You can use the Version History dialog box to view version information about the currently open test and to view or retrieve an earlier version of the test.

Viewing Version Information For a Test

You can view version information for any open test that has been stored in the Quality Center version control database, regardless of its current status.

To open the Version History dialog box for a test, open the test and choose **File > Quality Center Version Control > Version History**.



The Version History dialog box provides the following information:

Test name. The name of the currently open test.

Test status. The status of the test. The test can be:

- **Checked-in.** The test is currently checked in to the version control database. It is currently open in read-only format. You can check out the test to edit it.
- **Checked-out.** The test is checked out by you. It is currently open in read-write format.
- **Checked-out by <another user>.** The test is currently checked out by another user. It is currently open in read-only format. You cannot check out or edit the test until the specified user checks in the test.

My open version. The test version that is currently open on your QuickTest computer.

Version details. The version details for the test.

- **Version.** A list of all versions of the test.
- **User.** The user who checked in each listed version.
- **Date and Time.** The date and time that each version was checked in.

Version comments. The comments that were entered when the selected test version was checked in.

Working with Previous Test Versions

You can view an earlier version of a test in read-only mode, or you can check out an earlier version and then check it in as the latest version of the test.

To view an earlier version of a test:

1. Open the Quality Center test. The latest version of the test opens. For more information, see "Opening Tests from a Quality Center 9.x Project with Version Control Support" on page 1496.
2. Choose **File** > **Quality Center Version Control** > **Version History**. The Version History dialog box opens.

3 Select the test version you want to view in the **Version details** list.

4 Click the **Get Version** button. QuickTest reminds you that the test will open in read-only mode because it is not checked out.

5 Click **OK** to close the QuickTest message. The selected version opens in read-only mode.

Tips:

- To confirm the version number that you now have open in QuickTest, look at the **My open version** value in the Version History dialog box.
- After using the **Get Version** option to open an earlier version in read-only mode, you can check out the open test by choosing **File** > **Quality Center Version Control** > **Check Out**. This is the same as using the **Check Out** button in the Version History dialog box.

To check out an earlier version of a test:

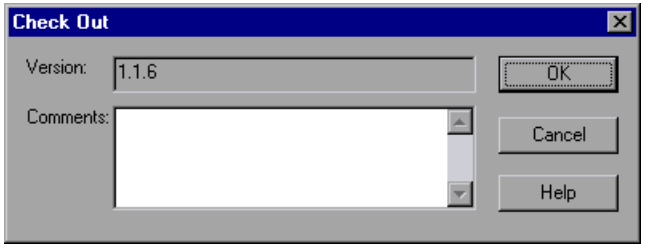
1 Open the Quality Center test. The latest version of the test opens. For more information, see "Opening Tests from a Quality Center 9.x Project with Version Control Support" on page 1496.

2 Choose **File** > **Quality Center Version Control** > **Version History**. The Version History dialog box opens.

3 Select the test version you want to view in the **Version details** list.

4 Click the **Check Out** button. A confirmation message opens.

5 Confirm that you want to check out an earlier version of the test. The Check Out dialog box opens and displays the test version to be checked out.



6 You can enter a description of the changes you plan to make in the **Comments** box.

7 Click **OK**. The open test closes and the selected version opens as a writable test.

8 View or edit the test as necessary.

9 If you want to check in your test as the latest version in the Quality Center database, choose **File** > **Quality Center Version Control** > **Check In**. If you do not want to upload the modified test to Quality Center, choose **File** > **Quality Center Version Control** > **Undo Check out**.

For more information on checking tests in, see "Checking Tests into the Version Control Database" on page 1499. For more information on undoing the check-out, see "Canceling a Check-Out Operation" on page 1504.

Canceling a Check-Out Operation

If you check out a test and then decide that you do not want to upload the modified test to Quality Center, you should cancel the check out operation so that the test will be available for check out by other Quality Center users.

To cancel a check out operation:

1 If it is not already open, open the checked out test.

2 Choose **File** > **Quality Center Version Control** > **Undo Check out**.

3 Click **Yes** to confirm the cancellation of your check out operation. The check out operation is cancelled. The checked out test closes, and the previously checked in version reopens in read-only mode.

Part XII

Working with Other HP Products

56

Working with Business Process Testing

When you are connected to a Quality Center project with Business Process Testing support, QuickTest enables you to create and/or implement the steps for the components that are used in Quality Center business process tests.

This chapter includes:

About Working with Business Process Testing

Business Process Testing enables Subject Matter Experts to create tests using a keyword-driven methodology for testing as well as an improved automated testing environment.

Business Process Testing integrates QuickTest with Quality Center and can be enabled by purchasing a specific Business Process Testing license. To work with Business Process Testing from within QuickTest, you must connect to a Quality Center project with Business Process Testing support.

This section provides an overview of the Business Process Testing model. For more information, see the *HP Business Process Testing User Guide* and the *HP QuickTest Professional for Business Process Testing User Guide*.

Understanding Business Process Testing Roles

The Business Process Testing model is role-based, allowing non-technical Subject Matter Experts (working in Quality Center) to collaborate effectively with Automation Engineers (working in QuickTest Professional). Subject Matter Experts define and document business processes, business components, and business process tests, while Automation Engineers define the required resources and settings, such as shared object repositories, function libraries, and recovery scenarios. Together, they can build, data-drive, document, and run business process tests, without requiring programming knowledge on the part of the Subject Matter Expert.

Note: The role structure and the tasks performed by various roles in your organization may differ from those described here according to the methodology adopted by your organization. These roles are flexible and depend on the abilities and time resources of the personnel using Business Process Testing. For example, the tasks of the Subject Matter Expert and the Automation Engineer may be performed by the same person. There are no product-specific rules or limitations controlling which roles must be defined in a particular organization, or which types of users can do which Business Process Testing tasks (provided that the users have the correct permissions).

The following user roles are identified in the Business Process Testing model:

Subject Matter Expert. The Subject Matter Expert has specific knowledge of the application logic, a high-level understanding of the entire system, and a detailed understanding of the individual elements and tasks that are fundamental to the application being tested. This enables the Subject Matter Expert to determine the operating scenarios or business processes that must be tested and identify the key business activities that are common to multiple business processes.

Using the Business Components module in Quality Center, the Subject Matter Expert creates business components that describe the specific tasks that can be performed in the application, and the condition or state of the application before and after those tasks. The Subject Matter Expert then defines the individual steps for each business component comprising the business process in the form of manual, or non-automated steps.

During the design phase, the Subject Matter Expert works with the Automation Engineer to identify the resources and settings needed to automate the components, enabling the Automation Engineer to prepare them. When the resources and settings are ready, the Subject Matter Expert automates the manual steps by converting them to keyword-driven components. Part of this process entails choosing an application area for each component. The application area contains all of the required resource files and settings that are specific to a particular area of the application being tested. Associating each component with an application area enables the component to access these resources and settings.

Using the Quality Center Test Plan module, the Subject Matter Expert combines the business components into business process tests, composed of a serial flow of the components. For example, most applications require users to log in before they can access any of the application functionality. The Subject Matter Expert could create one business component that represents this login procedure. This component procedure can be used in many business process tests, resulting in easier and more cost-efficient maintenance, updating, and test management.

The Subject Matter Expert configures the values used for business process tests, runs them in test sets, and reviews the results. The Subject Matter Expert is also responsible for maintaining the testing steps for each of the individual business components.

While defining components, Subject Matter Experts continue collaborating with the Automation Engineer. For example, they may request new operations (functions) for a component or discuss future changes planned for the component.

Automation Engineer. The Automation Engineer is an expert in using an automated testing tool, such as QuickTest Professional. The Automation Engineer works with the Subject Matter Expert to identify the resources that are needed for the various business process tests.

The Automation Engineer then prepares the resources and settings required for testing the features associated with each specific component, and stores them in an application area within the same Quality Center project used by the Subject Matter Experts who create and run the business process tests for the specific application.

Each application area serves as a single entity in which to store all of the resources and settings required for a component, providing a single point of maintenance for all elements associated with the testing of a specific part of an application. Application areas generally include one or more shared object repositories, a list of keywords that are available for use with a component, function libraries containing automated functions (operations), recovery scenarios for failed steps, and other resources and settings that are needed for a component to run correctly. Components are linked to the resources and settings in the application area. Therefore, when changes are made in the application area, all associated components are automatically updated.

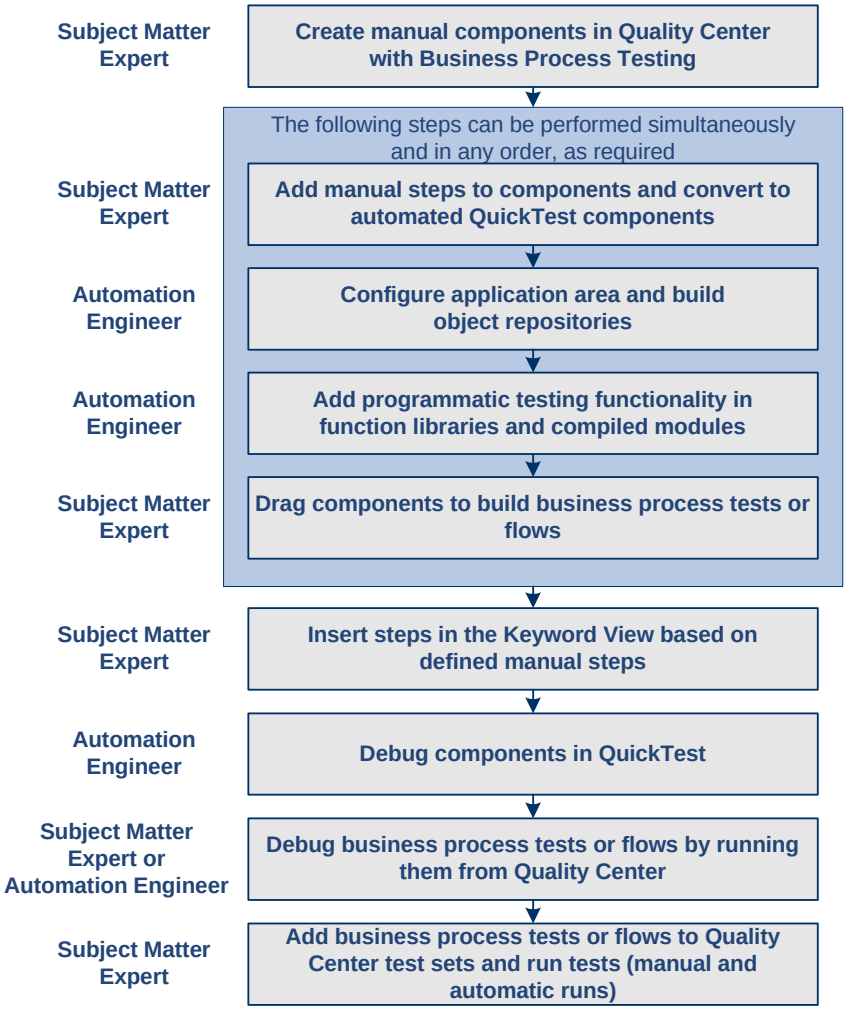
The Automation Engineer uses QuickTest features and functionality to create these resources from within QuickTest. For example, in QuickTest, the Automation Engineer can create and populate various object repositories with test objects that represent the different objects in the application being tested, even before the application is fully developed. The Automation Engineer can then add repository parameters, and so forth, as needed. The Automation Engineer can manage the various object repositories using the Object Repository Manager, and merge repositories using the Object Repository Merge Tool. Automation Engineers can also use QuickTest to create and debug function libraries containing functions that use programming logic to encapsulate the steps needed to perform a particular task.

Using the resources created by the Automation Engineer, the Subject Matter Experts can automate component steps, and create and maintain components and business process tests.

Automation Engineers can also create, debug, and modify components in QuickTest, if required.

Understanding the Business Process Testing Workflow

The following is an example of a common Business Process Testing workflow using QuickTest. The actual workflow in an organization may differ for different projects, or at different stages of the product development life cycle:



Understanding Business Process Testing Methodology

Each scenario that the Subject Matter Expert creates is a **business process test**. A business process test is composed of a serial flow of **components**. Each component performs a specific task. A component can pass data to a subsequent component.

Understanding Components

Components are easily-maintained reusable scripts that perform a specific task, and are the building blocks from which an effective business process testing structure can be produced. Components are parts of a business process that has been broken down into smaller parts. For example, in most applications users need to log in before they can do anything else. A Subject Matter Expert can create one component that represents the login procedure for an application. Each component can then be reused in different business process tests, resulting in easier maintenance, updating, and test management.

Components are comprised of steps. For example, the login component's first step may be to open the application. Its second step could be entering a user name. Its third step could be entering a password, and its fourth step could be clicking the **Enter** button.

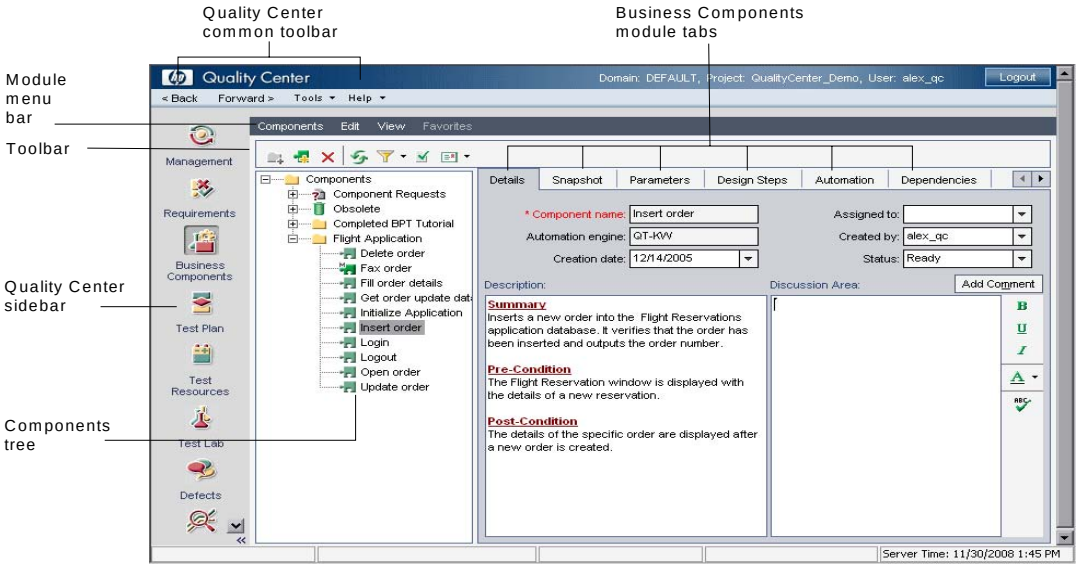
You can also add checkpoint steps and output values to your component.

- A **checkpoint** is a verification point that compares a current value for a specified property with the expected value for that property. This enables you to identify whether your application is functioning correctly. You can perform standard checkpoints and bitmap checkpoints on component steps. For more information, see "Understanding Checkpoints" on page 495.
- An **output value** is a step in which one or more values are captured at a specific point in your component and stored for the duration of the run session. The values can later be used as input at a different point in the run session. For more information, see "Outputting Values" on page 669.

You can create and edit components in QuickTest by adding steps on any supported environment, parameterizing selected items, and enhancing the component by incorporating functions (operations) that encapsulate the steps needed to perform a particular task. In Quality Center, a Subject Matter Expert creates components and combines them into business process tests, which are used to check that the application behaves as expected.

Creating Components in the Quality Center Business Components Module

The Subject Matter Expert can create a new component and define it in the Quality Center Business Components module.



The Business Components module includes the following tabs:

- **Details.** Provides a general summary of the component's purpose or goals, and the condition of the application before and after a component is run (its pre-conditions and post-conditions). You can specify details and implementation requirements for the currently selected business component.

- **Snapshot.** Displays an image that provides a visual cue or description of the component's purpose or operations. You can capture a snapshot image from the application and attach it to the currently selected business component.
- **Parameters.** Specifies the input and output component parameters and parameter values for the business component. Implementing and using parameters enables a component to receive data from an external source and to pass data to other components in the business process test flow.
- **Design Steps.** Enables you to create or view the manual steps of your business component, and to automate it if required.
- **Automation.** Displays or provides access to automated components. For keyword-driven components, enables you to create and modify the steps of your automated business component in a keyword-driven, table format, and provides a plain-language textual description of each step of the implemented component.
- **Dependencies tab.** Displays a list of assets that are linked to the currently selected business component.
- **History tab.** Displays a log of changes made to the component.

Implementing Components in QuickTest Professional

Generally, components are created by Subject Matter Experts in Quality Center, although they can also be created and debugged in QuickTest.

In QuickTest, you create components by adding steps manually—if the object repository is populated and the required operations are available. You can also create components by recording steps on any supported environment. You can parameterize selected items. You can also view and set options specific to components.

QuickTest enables you to create and modify two types of components: **business components** and **scripted components**. A business component is an easily-maintained, reusable unit comprising one or more steps that perform a specific task. A scripted component is an automated component that can contain programming logic. Scripted components share functionality with both test actions and business components.

For example, you can use the Keyword View, the Expert View, and other QuickTest tools and options to create, view, modify, and debug scripted components in QuickTest. Due to their complexity, scripted components can be edited only in QuickTest.

In Quality Center, the Subject Matter Expert can open components created in QuickTest. The Subject Matter Expert can then view and edit business components, but can only view the details for scripted components.

Creating Business Process Tests and Flows in the Quality Center Test Plan Module

The Subject Matter Expert first creates a business process test or flow in the Test Plan module. To populate the business process test or flow, the Subject Matter Expert then selects (drags and drops) the relevant components and configures their run settings.

Each component can be used differently by different business process tests or flows. For example, in each test the component can be configured to use different input parameter values or run a different number of iterations.

If, while creating a business process test or flow, the Subject Matter Expert realizes that a component has not been defined for an element that is necessary for the business process test or flow, the Subject Matter Expert can submit a component request from the Test Plan module.

Running Business Process Tests and Analyzing the Results

You can use the run and debug options in QuickTest to run and debug an individual component.

You can debug a business process test by running the test from the Test Plan module in Quality Center. When you choose to run from this module, you can choose which components to run in debug mode. (This pauses the run at the beginning of a component.)

When the business process test has been debugged and is ready for regular test runs, the Subject Matter Expert runs it from the Test Lab module similar to the way any other test is run in Quality Center. Before running the test, the Subject Matter Expert can define run-time parameter values and iterations using the **Iterations** column in the Test Lab module grid.

Note: When you run a business process test from Quality Center, the test run may also be influenced by settings in the QuickTest Remote Agent. For more information on the QuickTest Remote Agent, see "Setting QuickTest Remote Agent Preferences" on page 1441.

From the Test Lab module, you can view the results of the entire business process test run. The results include the value of each parameter, and the results of individual steps reported by QuickTest.

You can click the **Open Report** link to open the complete QuickTest report. The hierarchical report contains all the different iterations and components within the business process test run.

Understanding the Differences Between Components and Tests

If you are already familiar with using QuickTest to create action-based tests, you will find that the procedures for creating and editing components are quite similar. However, due to the design and purpose of the component model, there are certain differences in the way you create, edit, and run components. The guidelines below provide an overview of these differences.

- A component is a single entity. It cannot contain multiple actions or have calls to other actions or to other components.
- When working with components, all external resource files are stored in the Test Resources module of the Quality Center project to which you are currently connected.
- The name of the component node in the Keyword View is the same as the saved component. You cannot rename the node.
- Business components are created in the Keyword View, not the Expert View.
- You add resources via the component's application area, and not directly to the component.

57

Working with WinRunner

When you work with QuickTest, you can also run WinRunner tests and call TSL or user-defined functions in compiled modules.

This chapter includes:

About Working with WinRunner

If you have WinRunner installed on your computer, you can include calls to WinRunner tests and functions in your QuickTest test.

After you create a call to a WinRunner test or function, you can modify the argument values in call statements by editing them in the Expert View or Keyword View.

When QuickTest is connected to a Quality Center project that contains WinRunner tests or compiled modules, you can call a WinRunner test or function that is stored in that Quality Center project.

Calling WinRunner Tests

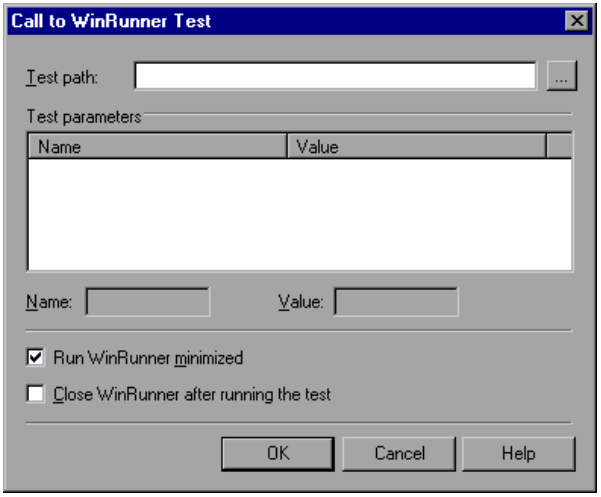
When QuickTest links to WinRunner to run a test, it starts WinRunner, opens the test, and runs it. Information about the WinRunner test run is displayed in the QuickTest Test Results window.

You can insert a call to a WinRunner test using the Call to WinRunner Test dialog box or by entering a **TSLTest.RunTestEx** statement in the Expert View.

Note: You cannot call a WinRunner test that includes calls to QuickTest tests.

To insert a call to a WinRunner test using the Call to WinRunner Test dialog box:

1 Select **Insert** > **Call to WinRunner** > **Test**. The Call to WinRunner Test dialog box opens.



2 In the **Test path** box, enter the path of the WinRunner test or browse to it.

3 The Parameters box lists any test parameters required for the WinRunner test. To enter values for the parameters:

- ➤ Highlight the parameter in the **Test Parameters** list. The selected parameter is displayed in the **Name** box below the list
- ➤ Enter the new value in the **Value** box.

Note: You can also use the parameter values from a QuickTest random number parameter, environment variable parameter, or from the QuickTest Data Table as the parameters for your WinRunner test. You do this by entering the parameter information manually in the TSLTest.RunTestEx statement. For more information, see "Passing QuickTest Parameterized Values to a WinRunner Test" on page 1520.

---

4 Select **Run WinRunner minimized** if you do not want to view the WinRunner window while the test runs.

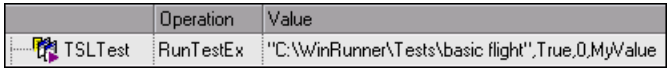
5 Select **Close WinRunner after running the test** if you want the WinRunner application to close when the step calling the WinRunner test is complete.

6 Click **OK** to close the dialog box.

For information on WinRunner test parameters, see the *HP WinRunner User's Guide*.

In QuickTest, the call to the WinRunner test is displayed as:

- ➤ a WinRunner **RunTestEx** step in the Keyword View. For example:



- ➤ a TSLTest.RunTestEx statement in VBScript in the Expert View. For example:

```
TSLTest.RunTestEx "C:\WinRunner\Tests\basic_flight",TRUE, 0, "MyValue"
```

The RunTestEx method has the following syntax:

TSLTest.RunTestEx *TestPath , RunMinimized, CloseApp [, Parameters]*

Note: Tests created in QuickTest 6.0 may contain calls to WinRunner tests using the RunTest method, which has slightly different syntax. Your tests will continue to run successfully with this method. However, it is recommended to update your tests to the **RunTestEx** method (and corresponding argument syntax). For more information on these methods, see the *HP QuickTest Professional Object Model Reference*.

After running the test, you can view the results. For more information, see "Viewing the Results" on page 1521.

For more information on the RunTestEx method and an example of usage, see the *HP QuickTest Professional Object Model Reference*.

Passing QuickTest Parameterized Values to a WinRunner Test

Rather than setting fixed values for the parameters required for a WinRunner test, you can pass WinRunner parameter values defined in a QuickTest Data Table, random or environment parameter. You specify these parameterized values by entering the appropriate statement as the *Parameters* argument in the TSLTest.RunTestEx statement.

For example, suppose you want to run a WinRunner test on a Windows-based Flight Reservation application, and that the test includes parameterized statements for the number of passengers on the flight and the seat class. You can pass the WinRunner test the value for its first parameter from a QuickTest random parameter (that generates a random number between 1 and 100), and pass it the value for the seat class from a QuickTest Data Table column labeled Class. Your TSLTest.RunTestEx statement in QuickTest might look something like this:

```
TSLTest.RunTestEx "D:\test1", TRUE, FALSE, RandomNumber(1, 100) ,
DataTable("Class", dtGlobalSheet)
```

For more information on the syntax and usage of the RandomNumber, Environment, and DataTable methods, see the Utility section of the *HP QuickTest Professional Object Model Reference*.

Viewing the Results

When you run a call to a WinRunner test, your QuickTest results include a node for each event that would normally be included in the WinRunner results. When you select a node corresponding to a WinRunner step, the right pane displays a summary of the WinRunner test and details about the selected step.

Note: You can also view the results of the called WinRunner test from the results folder of the WinRunner test. For WinRunner tests stored in Quality Center, you can also view the WinRunner test results from Quality Center.

For more information, see "Viewing WinRunner Test Steps in the Test Results" on page 1017.

For more information on designing and running WinRunner tests, see your WinRunner documentation.

Calling WinRunner Functions

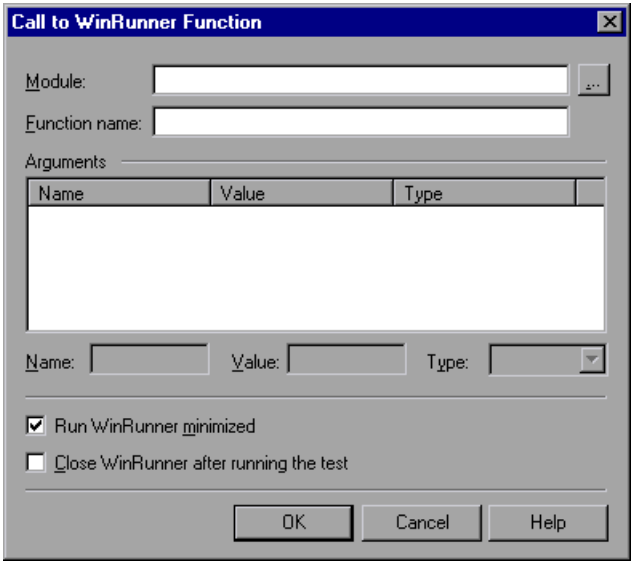
When QuickTest links to WinRunner to call a function, it starts WinRunner, loads the compiled module, and calls the function. This is useful when you want to use a user-defined function from WinRunner in QuickTest.

You call a WinRunner function from QuickTest by specifying the function and the compiled module containing the function.

Note: You cannot retrieve the values returned by the WinRunner function in your QuickTest test. However, you can view the returned value in the results.

To call a user-defined function from a WinRunner compiled module:

1 Select **Insert** > **Call to WinRunner** > **Function**. The Call to WinRunner Function dialog box opens.



2 In the **Module** box, enter the path of the compiled module containing the function or browse to it.

To call a WinRunner TSL function, enter the path of any compiled module.

3 In the **Function name** box, enter the name of a function defined in the specified compiled module, or enter any WinRunner TSL function.

4 Click inside the **Arguments** box. If WinRunner is currently open on your computer, the **Arguments** box displays the argument names as defined for the selected function. If WinRunner is not open, the **Arguments** box lists **p1**-**p15**, representing a maximum of fifteen (15) possible arguments for the function.

5 Enter values for **in** or **inout** arguments as follows:

- Highlight the argument in the **Arguments** box. The argument name is displayed in the **Name** box.
- If the argument type is "in" or "inout," enter the value in the **Value** box.
- In the **Type** box, select the correct argument type (**in**/**out**/**inout**).

Note: You can also use the parameter values from a QuickTest random or environment parameter or from the QuickTest Data Table as the **in** or **inout** arguments for your function. You do this by entering the argument information manually in the TSLTest.CallFuncEx statement. For more information, see "Passing QuickTest Parameters to a WinRunner Function" on page 1525.

For more information on function parameters, see the *HP WinRunner User's Guide*.

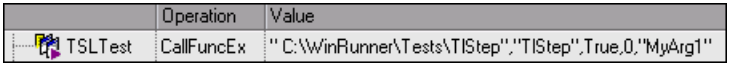
6 Select **Run WinRunner minimized** if you do not want to view the WinRunner window while the function runs.

7 Select **Close WinRunner after running the test** if you want the WinRunner application to close when the step calling the WinRunner function is complete.

8 Click **OK** to close the dialog box.

In QuickTest, the call to the TSL function is displayed as:

- a WinRunner **CallFuncEx** step in the Keyword View. For example:



- a TSLTest.CallFuncEx statement in VBScript in the Expert View. For example:

```
CallFuncEx "C:\WinRunner\Tests\TlStep","TlStep1",TRUE, 0, "MyArg1"
```

The CallFuncEx function has the following syntax:

TSLTest.CallFuncEx *ModulePath*, *Function*, *RunMinimized*, *CloseApp* [, *Arguments*]

Note: Tests created in QuickTest 6.0 may contain calls to WinRunner tests using the CallFunc method, which has slightly different syntax. Your tests will continue to run successfully with this method. However, it is recommended to update your tests to the **CallFuncEx** method (and corresponding argument syntax). For more information on these methods, see the *HP QuickTest Professional Object Model Reference*.

After running the test, you can view the results. For more information, see "Viewing the Results" on page 1525.

For information on WinRunner functions, function arguments, and WinRunner compiled modules, see the *HP WinRunner User's Guide* and the *HP WinRunner TSL Reference Guide*.

Passing QuickTest Parameters to a WinRunner Function

Rather than setting fixed values for the in and inout arguments in a WinRunner function, you can instruct QuickTest to have WinRunner use the parameter values defined in a QuickTest random or environment parameter, or in a QuickTest Data Table. You specify these parameters by entering the appropriate statement as the *Parameters* argument in the TSLTest.CallFuncEx statement.

For example, suppose you created a user-defined function in WinRunner that runs an application and enters the user name and password for the application.

You can instruct QuickTest to have WinRunner take the value for the user name and password from QuickTest Data Table columns labeled FlightUserName and FlightPwd. Your TSLTest.CallFuncEx statement in QuickTest might look something like this:

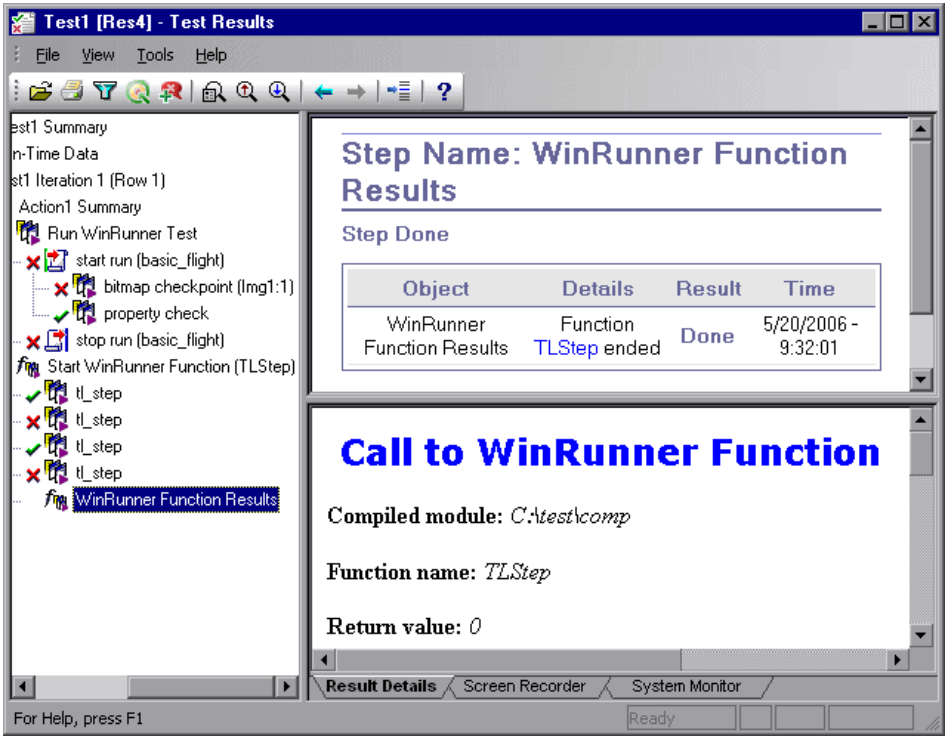
```
TSLTest.CallFuncEx "D:\flightfuncs", "run_flight", TRUE, FALSE,
DataTable("FlightUserName", dtGlobalSheet), DataTable("FlightPwd",
dtGlobalSheet)
```

For more information on the syntax and usage of the RandomNumber, Environment and DataTable methods, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

Viewing the Results

After you run a WinRunner function from QuickTest, you can view the results of your function call. The QuickTest Test Results window shows the start of the WinRunner function and the WinRunner function results. If the called function included events such as report_msg or tl_step, information about the results of these events are also included.

Highlight the **WinRunner Function Results** item in the results tree to display the function return value and additional information about the call to the function.



For more information on working with WinRunner functions and compiled modules, see your WinRunner documentation.

58

Working with HP Performance Testing and Business Availability Center Products

After you use QuickTest to create and run a suite of tests that test the functional capabilities of your application, you may want to test how much load your application can handle or to monitor your application as it runs.

HP performance testing products (LoadRunner and Performance Center) tests the performance of applications under controlled and peak load conditions. To generate load, These performance testing products run hundreds or thousands of virtual users. These virtual users provide consistent, repeatable, and measurable load to exercise your application just as real users would.

HP Business Availability Center enables real-time monitoring of the end user experience. Business Process Monitor runs virtual users to perform typical activities on the monitored application.

If you have already created and perfected a test in QuickTest that is a good representation of your users' actions, you may be able to use your QuickTest test as the basis for performance testing and application management activities. You can use Silent Test Runner to check in advance that a QuickTest test will run correctly from LoadRunner, Performance Center, and Business Process Monitor.

This chapter includes:

About Working with HP Performance Testing and Business Availability Center Products

QuickTest enables you to create complex tests that examine the full spectrum of your application's functionality to confirm that every element of your application works as expected in all situations.

The run mechanisms used in all HP Performance Testing and HP Business Availability Center products are the same. This means that you can create tests that are compatible with LoadRunner, Performance Center, and Business Process Monitor, enabling you to take advantage of tests or test segments that have already been designed and debugged in QuickTest.

For example, you can add QuickTest tests to specific points in a performance test to confirm that the application's functionality is not affected by the extra load at those sensitive points. You can also run QuickTest tests on Business Process Monitor to simulate end user experience and ensure that your application is running correctly and in a timely manner.

QuickTest also offers several features that are designed specifically for integration with LoadRunner, Performance Center, and Business Process Monitor. However, since these products are designed to run tests using virtual users representing many users simultaneously performing standard user operations, some QuickTest features may not be available when integrating these products with QuickTest.

If you do plan to use a single test in both QuickTest and LoadRunner, Performance Center, and/or Business Process Monitor, you should take into account the different options supported in each product as you design your test. For more information, see "Designing QuickTest Tests for Use with Performance Testing Products or Business Process Monitor" on page 1530 and "Inserting and Running Tests in a Performance Test or in Business Process Monitor" on page 1531.

Using QuickTest Performance Testing and Business Availability Center Features

You can use the Services object and its associated methods to insert statements that are specifically relevant to Performance Testing and Business Availability Center. These include AddWastedTime, EndDistributedTransaction, EndTransaction, GetEnvironmentAttribute, LogMessage, Rendezvous, SetTransaction, SetTransactionStatus, StartDistributedTransaction, StartTransaction, ThinkTime, and UserDataPoint. For more information on these methods, see the **Services** section of the *HP QuickTest Professional Object Model Reference* and your HP performance testing or Business Availability Center documentation.

You can also insert StartTransaction and EndTransaction statements using the **Insert** > **Start Transaction** and **Insert** > **End Transaction** menu options or toolbar buttons to insert the statement. For more information on these options, see "Measuring Transactions" on page 1534.

Note: LoadRunner, Performance Center, and Business Process Monitor use only the data that is included within a transaction, and ignore any data in a test outside of a transaction.

Designing QuickTest Tests for Use with Performance Testing Products or Business Process Monitor

The QuickTest tests you use with LoadRunner, Performance Center, or Business Process Monitor should be simple, designed to pinpoint specific operations, and should avoid using external actions and references to other external files (including resources stored in Quality Center). Also, when working with action iterations, corresponding StartTransaction and EndTransaction statements must be contained within the same action.

Designing Tests for Performance Testing

Consider the following guidelines when designing tests for use with performance testing products:

- Do not include references to external actions or other external resources (including resources stored in Quality Center), such as an external Data Table file, environment variable file, shared object repositories, function libraries, and so forth. This is because LoadRunner or Performance Center may not have access to the external action or resource. (However, if the resource can be found on the network, QuickTest will use it.)
- Every QuickTest test must contain at least one transaction to provide useful information in the performance test.
- Make sure that the last step(s) in the test closes the application being tested, as well as any child processes that are running. This enables the next iteration of the test to open the application again.

Designing Tests for Business Process Monitor

Consider the following guidelines when designing tests for use with Business Process Monitor:

- Every QuickTest test must contain at least one transaction to provide useful information in Business Process Monitor.
- When measuring a distributed transaction over two different Business Process Monitor profiles, the profile with the StartDistributedTransaction statement must be run before the profile with the associated EndDistributedTransaction.

- When measuring distributed transactions, make sure that you relate the tests to a single Business Process Monitor instance. Business Process Monitor searches for the end transaction name in all instances, and may close the wrong distributed transaction if it is included in more than one instance.
- When measuring a distributed transaction over two Business Process Monitor profiles, make sure that the timeout value you specify is large enough so that the profile that contains the StartDistributedTransaction step and all the profiles that run before the profile that contains the EndDistributedTransaction step, will finish running in a time that is less than the value of the specified timeout.
- Business Process Monitor does not support running QuickTest Professional tests that require access to external resources, including resources stored in Quality Center (such as a shared object repository, function library, external Data Table, external actions, and so forth). Tests that require external resources may fail to run on Business Process Monitor. (However, if the resource can be found on the network, QuickTest will use it.)
- Make sure that the last steps in the test close the application being tested, as well as any child processes that are running. This cleanup step enables the next test run to open the application again.

Inserting and Running Tests in a Performance Test or in Business Process Monitor

Before you insert and run your QuickTest test in a performance test or in Business Process Monitor, you should consider the guidelines below.

Note: You can simulate how the test will run from a performance test or from Business Process Monitor by using Silent Test Runner. For more information, see “Using Silent Test Runner” on page 1538.

Inserting and Running Tests in Performance Center and LoadRunner

- You can run only one GUI Vuser concurrently per computer. (A GUI Vuser is a Vuser that runs a QuickTest test.)

  To run multiple GUI Vusers on the same application you can open a terminal server session for each GUI Vuser. For more details refer to the HP performance testing documentation.

- To insert a QuickTest test in a LoadRunner scenario, in the Controller Open Test dialog box, browse to the test folder and select **QuickTest Tests** in the **Files of type** box (or select **Astra Tests** in LoadRunner versions older than 9.0). This enables you to view QuickTest tests in the folder.

- To use a QuickTest test in Performance Center, create a zipped version of the QuickTest test, and upload it to the Performance Center User Site Vuser Scripts Page.

- Ensure that QuickTest is closed on the QuickTest computer before running a QuickTest test in Performance Center or LoadRunner.

- Transaction breakdown is not supported for tests (scripts) created with QuickTest.

- QuickTest cannot run on a computer that is:

  - logged off or locked. In these cases, consider running QuickTest on a terminal server.

  - already running a QuickTest test. Make sure that the test is finished before starting to run another QuickTest test.

- The settings in the LoadRunner or Performance Center Run-time Settings dialog box are not relevant for QuickTest tests.

- You cannot use the **ResultDir** QuickTest environment variable when running a performance test.

  For more information on working with LoadRunner or Performance Center, see your HP performance testing documentation.

Inserting and Running Tests from Business Process Monitor

- Before you try to run a QuickTest test in Business Process Monitor, check which versions of QuickTest are supported by your version of Business Process Monitor. For more information, see the Business Process Monitor documentation.
- To run a QuickTest test in Business Process Monitor, QuickTest must be installed and closed on the Business Process Monitor computer
- Business Process Monitor can run only one QuickTest test at a time. Make sure that the previous QuickTest run session is finished before starting to run another QuickTest test.
- Transaction breakdown is not supported for tests created with QuickTest.
- QuickTest tests must be zipped before uploading them to Business Process Monitor.

  If you make changes to your local copy of a QuickTest test after uploading it to Business Availability Center, upload the zipped test again to enable Business Process Monitor to run the test with your changes.
- QuickTest cannot run tests on a computer that is logged off, locked, or running QuickTest as a non-interactive service.
- You cannot use the **ResultDir** QuickTest environment variable when running a test in Business Process Monitor.

  For more information on working with Business Availability Center, see your Business Availability Center documentation.

Measuring Transactions

You can measure how long it takes to run a section of your test by defining **transactions**. A transaction represents the process in your application that you are interested in measuring. Your test must include transactions to be used by LoadRunner, Performance Center, or the Business Process Monitor. These products use only the data that is included within a transaction, and ignore any data in a test outside of a transaction.

You define transactions within your test by enclosing the appropriate sections of the test with **start** and **end** transaction statements. For example, you can define a transaction that measures how long it takes to reserve a seat on a flight and for the confirmation to be displayed on the client's terminal.

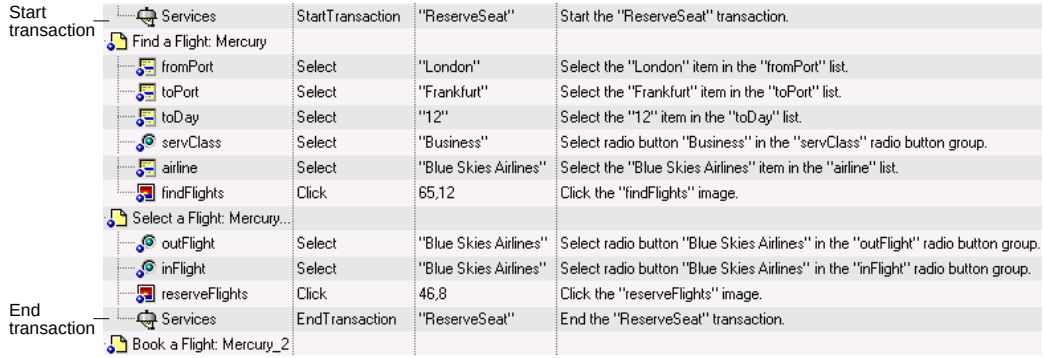
During the test run, the StartTransaction step signals the beginning of the time measurement. The time measurement continues until the EndTransaction step is reached. The test report displays the time it took to perform the transaction.

Note: If you start a transaction while there is already open transaction with the same name, the previous transaction is ended with **Fail** status and then the new transaction is started.

For information on the statements you can use in transactions, see the *HP QuickTest Professional Object Model Reference*.

There is no limit to the number of transactions that can be added to a test. You can also insert a transaction within a transaction.

Part of a sample test with a transaction is shown below, as it is displayed in the Keyword View:



Services	StartTransaction	"ReserveSeat"	Start the "ReserveSeat" transaction.
Find a Flight: Mercury			
fromPort	Select	"London"	Select the "London" item in the "fromPort" list.
toPort	Select	"Frankfurt"	Select the "Frankfurt" item in the "toPort" list.
toDay	Select	"12"	Select the "12" item in the "toDay" list.
servClass	Select	"Business"	Select radio button "Business" in the "servClass" radio button group.
airline	Select	"Blue Skies Airlines"	Select the "Blue Skies Airlines" item in the "airline" list.
findFlights	Click	65,12	Click the "findFlights" image.
Select a Flight: Mercury...			
outFlight	Select	"Blue Skies Airlines"	Select radio button "Blue Skies Airlines" in the "outFlight" radio button group.
inFlight	Select	"Blue Skies Airlines"	Select radio button "Blue Skies Airlines" in the "inFlight" radio button group.
reserveFlights	Click	46,8	Click the "reserveFlights" image.
Services	EndTransaction	"ReserveSeat"	End the "ReserveSeat" transaction.
Book a Flight: Mercury_2			

The same part of the test is displayed in the Expert View as follows:

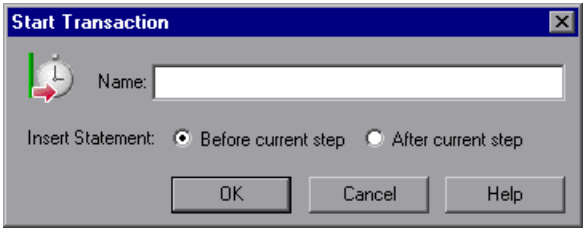
```
Services.StartTransaction "ReserveSeat"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("fromPort").Select "London"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("toPort").Select "Frankfurt"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("toDay").Select "12"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebRadioGroup("servClass").Select "Business"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("airline").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    Image("findFlights").Click 65,12
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").
    WebRadioGroup("outFlight").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").
    WebRadioGroup("inFlight").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").
    Image("reserveFlights").Click 46,8
Services.EndTransaction "ReserveSeat"
```

You can insert a variety of transaction-related statements using the Step Generator or Expert View. For more information, see the **Services** section of the *HP QuickTest Professional Object Model Reference*. You can also enter Start Transaction and End Transaction steps using options in the QuickTest window.

The Start Transaction Dialog Box

Description	Enables you to insert a step that signals the beginning of the time measurement for a transaction.
How to Access	➤ Select the **Insert** > **Start Transaction** menu command. ➤ Click the **Start Transaction** toolbar button .
Learn More	**Conceptual overview**: "Measuring Transactions" on page 1534 **Additional related topics**: "The End Transaction Dialog Box" on page 1537

Below is an image of the Start Transaction dialog box:



Start Transaction Dialog Box Options

Option	Description
Name	The name of the transaction you want to measure. **Note:** You cannot include spaces in a transaction name.
Insert Statement	Indicates where the **StartTransaction** step will be inserted in relation to the selected step. Select **Before current step** or **After current step**.

The End Transaction Dialog Box

Description	Enables you to insert a step that signals the end of the time measurement for a transaction.
How to Access	➤ Select the **Insert** > **End Transaction** menu command. ➤ Click the **End Transaction** toolbar button .
Important Information	There may be cases in which you want to instruct QuickTest to perform all the steps in a transaction, even though an error occurs during the run session. In the Run pane of the Test Settings dialog box (**File** > **Settings** > **Run** node), select **proceed to next step** from the **When error occurs during run session** list. You can also create recovery scenarios or other error handling steps to address these cases. For more information, see Chapter 48, "Defining and Using Recovery Scenarios."
Learn More	**Conceptual overview**: "Measuring Transactions" on page 1534 **Additional related topics**: "The Start Transaction Dialog Box" on page 1536

Below is an image of the End Transaction dialog box:

End Transaction Dialog Box Options

Option	Description
Name	The name of the transaction you want to end. The list contains the name of all transactions that start prior to the selected step in the current action.
Insert Statement	Indicates where the **EndTransaction** step will be inserted in relation to the selected step. Select **Before current step** or **After current step**.

Using Silent Test Runner

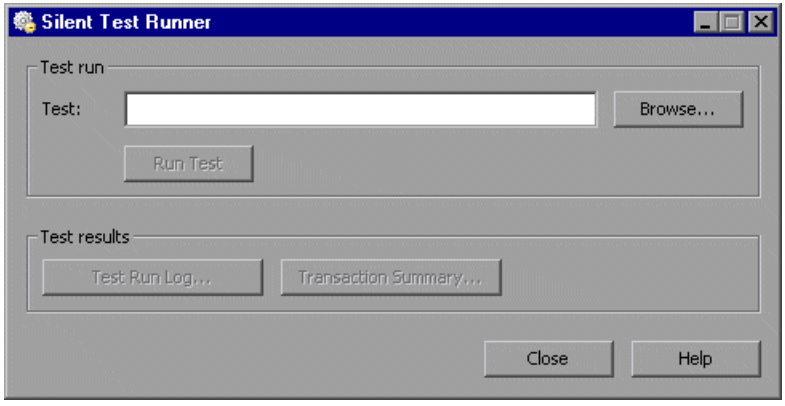
Silent Test Runner enables you to simulate the way a QuickTest test runs from LoadRunner, Performance Center, and Business Availability Center. When you run a test using Silent Test Runner, it runs without opening the QuickTest user interface, and the test runs at the same speed as when it is run from LoadRunner, Performance Center, or Business Availability Center At the end of the test run, you can view information about the test run and transaction times. For more information, see "Viewing Test Run Information for Silent Runs" on page 1541.

You can also use Silent Test Runner to verify that your QuickTest test is compatible with LoadRunner, Performance Center, and Business Availability Center. A test will fail when run using Silent Test Runner if it uses a feature that is not supported by these products. For more information on features that are not supported, see "Designing QuickTest Tests for Use with Performance Testing Products or Business Process Monitor" on page 1530, and "Inserting and Running Tests in a Performance Test or in Business Process Monitor" on page 1531.

The Silent Test Runner Dialog Box

Description	Enables you to simulate the way a QuickTest test runs from LoadRunner and Business Availability Center and to verify that your QuickTest test is compatible with LoadRunner and Business Availability Center.
How to Access	Select the **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Silent Test Runner** menu command.
Important Information	➤ You cannot run Silent Test Runner if QuickTest is already open or another test is currently running. You must close QuickTest and wait for its process to end before running your test using Silent Test Runner. ➤ You can invoke only one instance of Silent Test Runner and you can specify only one test to run. ➤ You cannot use the **ResultDir** QuickTest environment variable when running a test from Silent Test Runner.
Learn More	**Conceptual overview**: "Using Silent Test Runner" on page 1538 **Additional related topics**: "Viewing Test Run Information for Silent Runs" on page 1541

Below is an image of the Silent Test Runner dialog box:



Silent Test Runner Dialog Box Options

Option	Description
Test	The full file system path of the test you want to run. **Note**: To specify a network path, you must map the network drive.
Run Test	Enables you to run the test. (Enabled only when a test path is specified in the **Test** box). When you click this button, the test runs without opening the QuickTest user interface. The text **Running test...** is displayed next to the **Run Test** button while the test is running. When the test run finishes, the text **Running test...** is replaced with the text **Test run completed**. If Silent Test Runner was unable to run your test, the text **Test could not be run** is displayed. **Note:** After you start a test run, you cannot stop the test run from Silent Test Runner. Even if you close Silent Test Runner, the test continues to run. To end the run, end the **mdrv.exe** process manually.
Test Run Log	Enables you to view the most recent run log for the selected test. Each time you run a test with Silent Test Runner, the previous log file is overwritten with the current run results. (Enabled only when the selected test has run with the Silent Test Runner at least once.) For more information, see "Viewing the Test Run Log" on page 1541.
Transaction Summary	Enables you to view the summary of the transactions in the test. (Enabled only when the selected test contains at least one transaction and the test has run with the Silent Test Runner at least once.) For more information, see "Viewing the Transaction Summary" on page 1541.

Viewing Test Run Information for Silent Runs

Silent Test Runner provides test run information in log files. Each test generates a test run log, and any test with transactions generates an additional transaction summary.

Viewing the Test Run Log

The test run log is saved as **output.txt** in the **<QuickTest Professional>\Tests\<test name>** folder. A log file is saved for each test run with Silent Test Runner and is overwritten when you rerun the test. To open the log file, click **Test Run Log**.

The log file displays information about the test run. For example, information is shown about each iteration, action call, step transaction, failed step, and so forth. Each line displays a message or error ID. For more information on message and error codes in the log file, see your Performance Center or Business Availability Center documentation.

Viewing the Transaction Summary

The transaction summary is saved as **transactions.txt** in the **<QuickTest Professional>\Tests\<test name>** folder. A transaction summary is saved for each test that includes transactions and is overwritten when you rerun the test. To open the log file, click **Transaction Summary**. The transaction summary displays a line for each transaction in the test. For each transaction, the status is displayed together with the total duration time and any wasted time (in seconds). The transaction measurements in Silent Test Runner are exactly the same as if the test was run from LoadRunner, Performance Center, or Business Availability Center.

Notes:

- A transaction summary is available only for a test that contains transactions ending with an EndTransaction statement. If a transaction started but did not end because of test failure, it is not included in the transaction summary.
- Distributed transactions (transactions that start in one test and end in another) are not reported in the transaction summary but are included in the test run log.
- Any transaction information included in the transaction summary is also included in the test run log.

Part XIII

Appendixes

A

Supported Checkpoints and Output Values Per Add-In

The tables in this chapter show the categories of checkpoints and output values that are supported by QuickTest Professional for each add-in.

For more information about using checkpoints and output values in a specific add-in, see the relevant add-in section.

This chapter includes:

Supported Checkpoints

Table Legend

- S: Supported
- NS: Not Supported
- NA: Not Applicable

For additional information, see "Footnotes" on page 1547.

	Accessibility	Bitmap	Database	Image	Page	Standard	Table	Text	Text Area	XML (From Application)	XML (From Resource)
ActiveX	NS	S	NA	NS	NA	S	S	S	S	NA	NS
Delphi	NS	S	NA	NS	NA	S	S	S	S	NA	S
Java	NA	S	NA	NA	NA	S	S	S[6]	S	NA	NS
.NET Web Forms[5]	S	S	NA	NA	NA	S	S	S	NS	S	S
.NET Windows Forms	NA	S	NA	NA	NA	S	S	N	S	N	N
Oracle	NA	S	NA	NA	NA	S	S	NS	NS	NA	NA
PeopleSoft	S	S	NA	S	S	S	S	S[3]	NS	S	S
PowerBuilder[4]	NS	S	NA	NS	NA	S	S	S	S	NA	NS
SAP Web	S	S	NA	S	S	S	S	S	NS	S	S
SAP Windows	S[7]	S	NA	S[7]	S[7]	S	S	S[7]	NS	S[7]	NA
Siebel	S	S	NA	S	S	S	S	S	NS	S	S
Standard Windows	NS	S	NA	NS	NA	S	S	S	S	NA	NS
Stingray	NA	S	NA	NA	NA	S	S	S	S	NA	NS
Terminal Emulator	NA	S	NA	NA	NA	S	NA	NA	NA	NA	NA

	Accessibility	Bitmap	Database	Image	Page	Standard	Table	Text	Text Area	XML (From Application)	XML (From Resource)
Visual Age	NA	S	NA	NA	NA	S	S	S	S	NA	NS
Visual Basic	NS	S	NA	NS	NA	S	S	S	S	NA	NS
Web[2]	S	S	NA	S	S	S	S	S[3]	NS	S	NS
Web Services	NA	NA	NA	NA	NA	S	NA	NA	NA	S	NS
WPF	NA	S	NA	NA	NA	S	NA	S	S	NA	NA

Footnotes

[1] Only standard and bitmap checkpoints are supported for business components.

[2] When creating checkpoints for Web objects in components, only bitmap checkpoints and standard checkpoints are available.

[3] Checkpoints are supported only for Page, Frame, and ViewLink objects.

[4] When you insert a checkpoint on a PowerBuilder DataWindow control, QuickTest treats it as a table and opens the Table Checkpoint Properties dialog box (not supported for components).

[5] For NET Web Forms, text checkpoints for WbfTreeView, WbfToolbar, and WbfTabStrip objects are not supported.

[6] The text checkpoint mechanism for Java objects is disabled by default. You can enable it (for tests only) in the Advanced Java Options dialog box.

[7] This is supported only when QuickTest records HTML elements using the Web infrastructure, but not when it records using the SAPGui Scripting Interface (as selected in the SAP pane of the Options dialog box).

Supported Output Values

Table Legend

- S: Supported
- NS: Not Supported
- NA: Not Applicable

For additional information, see "Footnotes" on page 1549.

	Accessibility	Bitmap	Database	Page	Standard	Table	Text	Text Area	XML (From Application)	XML (From Resource)
ActiveX	NS	NA	NA	NA	S	S	S	S	NA	S
Delphi	NS	NA	NA	NA	S	NA	S	S	NA	S
Java	NA	NA	NA	NA	S	NA	S[6]	NA	NA	NA
NET Web Forms	NA	NA	NA	S	S	S	S	NA	NA	NA
NET Windows Forms	NA	NA	NA	NA	S	S	NA	NA	NA	NA
Oracle	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
PeopleSoft	NA	NA	NA	S	S	S	S[3]	NS	S	S
PowerBuilder[4]	NA	NA	NA	NA	S	NA	S	S	NA	S
SAP Web	NA	NA	NA	S	S	S	S	NS	S	S
SAP Windows	NA	NA	NA	S[6]	S	S	S[6]	NS	S[6]	S
Siebel	NA	NA	NA	S	S	S	S	NS	S	S
Standard Windows	NA	NA	NA	NA	S	NA	S	S	NA	S
Stingray	NA	NA	NA	NA	S	NA	S	S	NA	S
Terminal Emulator	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

	Accessibility	Bitmap	Database	Page	Standard	Table	Text	Text Area	XML (From Application)	XML (From Resource)
Visual Age	NA	NA	NA	NA	NA	S	S	S	NA	NA
Visual Basic	NA	NA	NA	NA	S	NA	S	S	NA	S
Web[2]	NA	NA	NA	S	S	S	S[3]	NS	S	NA
Web Services	NA	NA	NA	NA	NA	NA	NA	NA	NA	S
WPF	NA	NA	NA	NA	S	NA	S	S	NA	NA

Footnotes

[1] Only standard and bitmap output values are supported for business components.

[2] When creating output values for Web objects in components, only standard output values are available.

[3] Output values are supported only for Page, Frame, and ViewLink objects.

[4] When you insert an output value step on a PowerBuilder DataWindow control, QuickTest treats it as a table and opens the Table Output Value Properties dialog box (not supported for components).

[5] The text output mechanism for Java objects is disabled by default. You can enable it (for tests only) in the Advanced Java Options dialog box.

[6] This is supported only when QuickTest records HTML elements using the Web infrastructure, but not when it records using the SAPGui Scripting Interface (as selected in the SAP pane of the Options dialog box).

B

Frequently Asked Questions

This chapter answers some of the questions that are asked most frequently by advanced users of QuickTest. The questions and answers are divided into the following sections:

This chapter includes:

Creating Tests

➤ **How can I record on objects or environments not supported by QuickTest?**

You can do this in a number of ways:

- ➤ Install and load any of the add-ins that are available for QuickTest Professional. QuickTest supports many developmental environments including Java, Oracle, .NET, SAP Solutions, Siebel, PeopleSoft, terminal emulators, and Web services.
- ➤ You can map objects of an unidentified or custom class to standard Windows classes. For more information on object mapping, see "Mapping User-Defined Test Object Classes" on page 131.
- ➤ QuickTest provides add-in extensibility that you can use to extend QuickTest built-in support for various objects. This enables you to direct QuickTest to recognize an object as belonging to a specific test object class, and to specify the behavior of the test object. You can also extend the list of available test object classes that QuickTest recognizes. This enables you to create tests that fully support the specific behavior of your custom objects.
- ➤ You can define **virtual objects** for objects that behave like test objects, and then record in the normal recording mode. For more information on defining virtual objects, see Chapter 47, "Learning Virtual Objects."
- ➤ You can record your clicks and keyboard input based on coordinates in the **low-level recording** or **analog** modes. For more information on low-level and analog recording, see "Choosing the Recording Mode" on page 368.

➤ **How can I launch an application from a test?**

An application can be launched from within a test by adding a **SystemUtil** step to your test, such as:

```
SystemUtil.Run "D:\My Music\Breathe.mp3","","D:\My Music\Details","open"
```

For Windows-based applications, you should also ensure that in the Windows Applications tab of the Record and Run Settings dialog box, you configure QuickTest to record and run on applications opened by QuickTest.

➤ **How does QuickTest capture user processes in Web pages?**

QuickTest hooks the Microsoft Internet Explorer browser. As the user navigates the Web-based application, QuickTest records the user operations. (For information on modifying which user operations are recorded, see the section on configuring Web event recording in the *HP QuickTest Professional Add-ins Guide*.) QuickTest can then run the test by running the steps as they originally occurred.

Programming in the Expert View

➤ **Can I store functions and subroutines in a function library?**

You can define functions within an individual action, or you can create one or more VBScript function libraries containing your functions, and then call them from any test. You can use the QuickTest function library editor to create and debug your function libraries.

You can also register your functions as methods for QuickTest test objects. Your registered methods can override the functionality of an existing test object method for the duration of a run session, or you can register a new method for a test object class.

For more information, see Chapter 31, "Working with User-Defined Functions and Function Libraries".

You can help improve QuickTest performance by storing your functions in function libraries instead of as reusable actions.

➤ **How can I enter information during a run session?**

The VBScript InputBox function enables you to display a dialog box that prompts the user for input and then continues running the test. You can use the value that was entered by the user later in the run session. For more information on the InputBox function, see the *VBScript Reference*.

The following example shows the InputBox function used to prompt the user for a password:

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set
"administrator"
Passwd = InputBox ("Enter password", "User Input")
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("password").Set
Passwd
```

➤ **I have a Microsoft Access database that contains data I would like to use in my test. How do I do this?**

The Expert View enables you to access databases using ADO and ODBC. Below is a sample test that searches for books written by an author in the "Authors" table of the database.

```
Dim MyDB
Dim MyEng
Set MyEng = CreateObject("DAO.DBEngine.35")
Dim Td
Dim rs

' Specify the database to use.
Set MyDB = MyEng.OpenDatabase("BIBLIO.MDB")

' Read and use the name of the first 10 authors.
Set Td = MyDB.TableDefs("Authors")
Set rs = Td.OpenRecordset
rs.MoveFirst
For i = 1 To 10
    Browser("Book Club").Page("Search Books").WebEdit("Author Name").Set
rs("Author")
    Browser("Book Club").Page("Search Books").WebButton("Search").Click
Next
```

➤ **How do I customize the Test Results?**

You can add information to the test results report by using the **ReportEvent** method, for example:

```
Reporter.ReportEvent 1, "Custom Step", "The user-defined step failed"
```

For more information, see the *HP QuickTest Professional Object Model Reference*.

The results of each QuickTest run session are saved in a single **.xml** file (called **results.xml**). You can modify this file, as needed. You can use the **QuickTest Test Results Schema** (available from the QuickTest Professional Help) to help you customize your test results.

Working with Dynamic Content

➤ **How can I create and run tests on objects that change dynamically from viewing to viewing?**

Sometimes the content of objects in an application changes due to dynamic content. You can create dynamic descriptions of these objects so that QuickTest will recognize them when it runs the test using regular expressions, the **Description** object, repository parameters, or **SetTOProperty** steps.

➤ **How can I check that a child window exists (or does not exist)?**

Sometimes a link in one window creates another window.

You can use the **Exist** property to check whether or not a window exists. For example:

```
If Window("Main").ActiveX("Slider").Exist Then
. . .
```

You can also use the ChildObjects method to retrieve all child objects (or the subset of child objects that match a certain description) on the Desktop or within any other parent object.

Example:

```
Set oDesc = Description.Create
oDesc("Class Name").Value = "Window"

Set coll = Desktop.ChildObjects(oDesc)
For i = 0 to coll.count -1
        msgbox coll(i).GetROProperty("text")
Next
```

For more information on the Exist property and ChildObjects method, see the *HP QuickTest Professional Object Model Reference*.

➤ **How does QuickTest record on dynamically generated URLs and Web pages?**

QuickTest actually clicks links as they are displayed on the page. Therefore, QuickTest records how to find a particular object, such as a link on the page, rather than the object itself. For example, if the link to a dynamically generated URL is an image, then QuickTest records the "IMG" HTML tag, and the name of the image. This enables QuickTest to find this image in the future and click on it.

➤ **How does QuickTest handle tabs in browsers?**

QuickTest provides several methods that you can use with the **Browser** test object to manage tabs in your Web browser.

OpenNewTab opens a new tab in the current Web browser.

IsSiblingTab indicates whether a specified tab is a sibling of the current tab object in the same browser window.

Close closes the current tab if more than one tab exists, and closes the browser window if the browser contains only one tab.

CloseAllTabs closes all tabs in a browser and closes the browser window.

For more information on these **Browser**-related methods, see the **Web** section of the *HP QuickTest Professional Object Model Reference*.

Advanced Web Issues

➤ **How does QuickTest handle cookies?**

Server side connections, such as CGI scripts, can use cookies both to store and retrieve information on the client side of the connection.

QuickTest stores cookies in the memory for each user, and the browser handles them as it normally would.

➤ **Where can I find a Web page's cookie?**

The cookie used by the Internet Explorer browser can be accessed through the browser's Document Object Model (DOM) using the **.Object** property. In the following example the cookie collection is returned the from the browser:

```
Browser("Flight reservations").Page("Flight reservations").Object.Cookie
```

➤ **How does QuickTest handle session IDs?**

The server, not the browser, handles session IDs, usually by a cookie or by embedding the session ID in all links. This does not affect QuickTest.

➤ **How does QuickTest handle server redirections?**

When the server redirects the client, the client generally does not notice the redirection, and misdirections generally do not occur. In most cases, the client is redirected to another script on the server. This additional script produces the HTML code for the subsequent page to be viewed. This has no effect on QuickTest or the browser.

➤ **How does QuickTest handle meta tags?**

Meta tags do not affect how the page is displayed. Generally, they contain information only about who created the page, how often it is updated, what the page is about, and which keywords represent the page's content. Therefore, QuickTest has no problem handling meta tags.

➤ **Does QuickTest work with .asp and .jsp?**

Dynamically created Web pages utilizing Active Server Page technology have an .asp extension. Dynamically created Web pages utilizing Java Server Page technology have a .jsp extension. These technologies are completely server-side and have no bearing on QuickTest.

➤ **How does QTP support AJAX?**

You can use QuickTest Professional Web Add-in Extensibility to add your own support for custom Web controls. The Web Add-in Extensibility SDK installs a sample toolkit support set that provides partial support for some ASP .NET AJAX controls. You can use this sample to learn how to create your own support for your AJAX controls. For more information, see the *HP QuickTest Professional Web Add-in Extensibility Developer Guide*.

➤ **Does QuickTest work with COM?**

QuickTest complies with the COM standard.

QuickTest supports COM objects embedded in Web pages (which are currently accessible only using Microsoft Internet Explorer), and you can drive COM objects in VBScript.

➤ **Does QuickTest work with XML?**

XML is eXtensible Markup Language, a pared-down version of SGML for Web documents, that enables Web designers to create their own customized tags. QuickTest supports XML and recognizes XML tags as objects.

You can also create XML checkpoints to check the content of XML documents in Web pages, frames or files. QuickTest also supports XML output and schema validation.

For more information, see Chapter 23, "Checking XML," and the **XMLUtil** object in the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

➤ **How can I access HTML tags directly?**

QuickTest provides direct access to the Internet Explorer's Document Object Model (DOM) through which you can access the HTML tags directly. Access to the DOM is performed using the .Object notation.

The test below demonstrates how to iterate over all the tags in an Internet Explorer page. The test then outputs the inner-text of the tags (the text contained between the tags) to the Test Results using the Reporter object.

```
' Use the on error option because not all the elements have inner-text.
On Error Resume Next
Set Doc = Browser("CNN Interactive").Page("CNN Interactive").Object

' Loop through all the objects in the page.
For Each Element In Doc.all
    TagName = Element.TagName ' Get the tag name.
    InnerText = Element.innerText ' Get the inner text.

    ' Write the information to the test results.
    Reporter.ReportEvent 0, TagName, InnerText
Next
```

➤ **Where can I find information on the Internet Explorer Document Object Model?**

For information on the Internet Explorer DOM, browse to the following Web sites:

Document object:
http://msdn2.microsoft.com/en-us/library/ms531073.aspx

Other DHTML objects:
http://msdn2.microsoft.com/en-us/library/ms533054.aspx

General DHTML reference:
http://msdn2.microsoft.com/en-us/library/ms533050.aspx

➤ **How can I send keyboard key commands (such as shortcut commands) to objects that do not support the Type method?**

For objects that do not support the **Type** method, use the Windows Scripting **SendKeys** method. For more information, see the Microsoft VBScript Language Reference (choose **Help** > **QuickTest Professional Help** > **VBScript Reference** > **Windows Script Host**).

Standard Windows Environment

➤ **How can I record on nonstandard menus?**

You can modify how QuickTest behaves when it records menus. The options that control this behavior are located in the Windows Applications > Advanced Options pane.
(**Tools** > **Options** > **Windows Applications** node> **Advanced** node).

For more information, see the *HP QuickTest Professional Add-ins Guide*.

➤ **How can I terminate an application that is not responding?**

You can terminate any standard application while running a test in QuickTest by adding one of the following steps to the test:

- SystemUtil.CloseProcessByName "app.exe"
- SystemUtil.CloseProcessByWndTitle "Some Title"

➤ **Can I copy and paste to and from the Clipboard during a run session?**

You can use the Clipboard object to copy, cut, and paste text during a QuickTest run session.

The Clipboard object supports the same methods as the Clipboard object available in Visual Basic, such as:

- Clear
- GetData
- GetText
- SetData
- SetText

For more information on these methods, see http://msdn.microsoft.com/en-us/library/ms172962.aspx.

Below is an example of Clipboard object usage:

```
Set MyClipboard = CreateObject("Mercury.Clipboard")
MyClipboard.Clear
MyClipboard.SetText "TEST"
MsgBox MyClipboard.GetText
```

Test Maintenance

➤ **How do I maintain my test when my application changes?**

The way to maintain a test when your application changes depends on how much your application changes. This is one of the main reasons you should create a small group of tests rather than one large test for your entire application.

You can also use QuickTest actions to design more modular and efficient tests. Divide your test into several actions, based on functionality. When your application changes, you can modify a specific action, without changing the rest of the test. Whenever possible, insert calls to reusable actions rather than creating identical pieces of script in several tests. This way, changes to your original reusable action are automatically applied to all tests calling that action. For more information, see Chapter 16, "Working with Advanced Action Features."

If you have many tests and actions that contain the same test objects, it is recommended to work with shared object repositories so that you can update object information in a centralized location.

You can use the **Update Run Mode** option to update changed information for checkpoints or the Active Screen, or to change the set of identification properties used to identify the objects in your application. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.

If there is a discrepancy between the identification property values saved in the object repository and the object property values in the application, you can use the **Maintenance Run Mode** to help correct this. When you run a test in Maintenance Run Mode, QuickTest runs your test, and then guides you through the process of updating your steps and object repository each time it encounters a step it cannot perform due to an object repository discrepancy. For more information, see "Running Tests with the Maintenance Run Wizard" on page 1104.

➤ **Can I increase or decrease Active Screen information after I finish recording a test?**

If you find that the information saved in the Active Screen after recording is not sufficient for your test editing needs, or if you no longer need Active Screen information, and you want to decrease the size of your test, there are several methods of changing the amount of Active Screen information saved with your test.

- To decrease the disk space used by your test, you can delete Active Screen information by selecting **Save As**, and clearing the **Save Active Screen files** check box. For more information, see "Saving a Test" on page 324.
- If you chose not to save all information in the Active Screen when testing a Windows application, you can use one of several methods to increase the information stored in the Active Screen.

  Confirm that the Active Screen capture preference in the Active Screen pane of the Options dialog box is set to capture the amount of information you need and then:

  - Perform an **Update Run Mode** operation to save the required amount of information in the Active Screen for all existing steps. For more information on the **Update Run Mode** options, see "Updating a Test Using the Update Run Mode Option" on page 1125.
  - Re-record the steps containing the objects you want to add to the Active Screen.

    To re-record the step, select the step after which you want to record your step, position your application to match the selected location in your test, and then begin recording. Alternatively, place a breakpoint in your test at the step before which you want to add a step and run your test to the breakpoint. This brings your application to the point from which to record the step. For more information on setting breakpoints, see "Setting Breakpoints" on page 1079.

For more information on changing the amount of information saved in the Active Screen for Windows applications, see "Setting Active Screen Options" on page 1240.

➤ **How can I remove test result files from old tests?**

You can use the Test Results Deletion Tool to view a list of all of the test results in a specific location in your file system or in your Quality Center project. You can then delete any test results that you no longer require.

The Test Results Deletion Tool enables you to sort the test results by name, date, size, and so forth, so that you can more easily identify the results you want to delete.

To open this utility, choose **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Test Results Deletion Tool**.

Testing Localized Applications

➤ **I am testing localized versions of a single application, each with localized user interface strings. How do I create efficient tests in QuickTest?**

You can parameterize these user interface strings using parameters from the global Environment variable list. This is a list of variables and corresponding values that can be accessed from any test. For more information, see Chapter 24, "Parameterizing Values."

➤ **I am testing localized versions of a single application. How can I efficiently input different data in my tests, depending on the language of the application?**

If you are running a single iteration of your test, or if you want values to remain constant for all iterations of an action or test, use environment variables, and then change the active environment variable file for each test run.

If you are running multiple iterations of your test or action, and you want the input data to change in each iteration, you can create an external Data Table for each localized version of your application. When you change the localized version of the application you are testing, you simply switch the Data Table file for your test in the Resources pane of the Test Settings dialog box. For more information on working with Data Tables, see Chapter 42, "Working with Data Tables." For more information on selecting the Data Table file for your test, see "Defining Resource Settings for Your Test" on page 1274.

Improving QuickTest Performance

How can I improve the working speed of QuickTest?

You can improve the working speed of QuickTest by doing any of the following:

- In the Add-in Manager, load only the add-ins you need for a specific QuickTest session when QuickTest starts. This will improve performance while learning objects and during run sessions. For more information on loading add-ins, see the *HP QuickTest Professional Add-ins Guide*.
- Minimize the number of actions in a test. Ideally, a test should not contain more than a few dozen actions.
- Store your functions in function libraries instead of as reusable actions.
- Run your tests in "fast mode." From the Run pane in the Options dialog box, select the **Fast** option. This instructs QuickTest to run your test without displaying the execution arrow for each step, enabling the test to run faster. For more information on the Run pane of the Options dialog box, see “Setting Run Testing Options” on page 1253.
- If you are not using the Active Screen while editing your test, hide the Active Screen while editing your test to improve editing response time. Choose **View** > **Active Screen**, or toggle the Active Screen toolbar button to hide the Active Screen. For more information, see Chapter 2, “QuickTest at a Glance.”

- Decide if and how much information you want to capture and save in the Active Screen. The more information you capture, the easier it is to add steps to your test using the many Active Screen options, but more captured information also leads to slower recording and editing times. You can choose from the following Active Screen options to improve performance:
  - If you are testing Windows applications, you can choose to save all Active Screen information in every step, save information only in certain steps, or to disable Active Screen captures entirely. You set this preference in the Active Screen pane of the Options dialog box. For more information, see "Setting Active Screen Options" on page 1240.
  - If you are testing Web applications, you can disable screen capture of all steps in the Active Screen. From the Active Screen pane of the Options dialog box, click **Custom Level** to open the Custom Active Screen Capture Settings dialog box.

    Select the **Disable Active Screen Capture** option. This will improve recording time. For more information on the Active Screen pane of the Options dialog box, see "Setting Active Screen Options" on page 1240.
  - When you save a new test, or when you save a test with a new name using **Save As**, you can choose not to save the captured Active Screen files with the test by clearing the **Save Active Screen files** option in the Save or Save As dialog box. This is especially useful when you have finished designing your test and you plan to use your test only for test runs. Tests without Active Screen files open more quickly and use significantly less disk space.

    For more information on the Active Screen pane of the Options dialog box, see "Setting Active Screen Options" on page 1240.

Tip: If you need to recover Active Screen files after you save a test without Active Screen files, re-record the necessary steps or use the **Update Run Mode** option to recapture screens for all steps in your test. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.

- Decide if and when you want to capture and save images and/or movies of the application for the test results. You can reduce disk space and improve test run time by saving screen captures and movie segments only in certain situations, such as when errors occur, or by not saving them at all. To do this, use the **Save still image captures to results** and **Save movie to results** options in the Run > Screen Capture pane in the Options dialog box. For more information, see "The Options Dialog Box: Run > Screen Capture Pane" on page 1255.
- Save the test results report to a temporary folder to overwrite the results from the previous run session every time you run a test. For more information, see "Running Your Entire Test" on page 955.
- Use the Results Deletion Tool to remove unwanted or obsolete test results from your system, according to specific criteria that you define. This enables you to free up valuable disk space. For more information, see "Deleting Results Using the Test Results Deletion Tool" on page 1004.

How can I decrease the disk space used by QuickTest?

You can decrease the disk space used by QuickTest by doing any of the following:

- Decide if and when you want to capture and save images and/or movies of the application for the test results. You can reduce disk space and improve test run time by saving screen captures and movie segments only in certain situations, such as when errors occur, or by not saving them at all. To do this, use the **Save still image captures to results** and **Save movie to results** options in the Run > Screen Capture pane in the Options dialog box. For more information, see "The Options Dialog Box: Run > Screen Capture Pane" on page 1255.

- Decide if and how much information you want to capture and save in the Active Screen. The more information you capture, the easier it is to add steps to your test using the many Active Screen options, but more captured information also leads to slower recording and editing times. You can choose from the following Active Screen options to improve performance:
  - If you are testing Windows applications, you can choose to Save all Active Screen information in every step, save information only in certain steps, or to disable Active Screen captures entirely. You set this preference in the Active Screen pane of the Options dialog box. For more information, see "Setting Active Screen Options" on page 1240.
  - If you are testing Web applications, you can disable screen capture of all steps in the Active Screen. From the Active Screen pane, click **Custom Level** to open the Custom Active Screen Capture Settings dialog box. Select the **Disable Active Screen Capture** option. This will improve recording time. For more information on the Active Screen pane of the Options dialog box, see "Setting Active Screen Options" on page 1240.
  - When you save a new test, or when you save a test with a new name using Save As, you can choose not to save the captured Active Screen files with the test by clearing the **Save Active Screen files** option in the Save or Save As dialog box. This is especially useful when you have finished designing your test and you plan to use your test only for test runs. Tests without Active Screen files use significantly less disk space.

Tip: If you need to recover Active Screen files after you save a test without Active Screen files, re-record the necessary steps or use the **Update Run Mode** option to recapture screens for all steps in your test. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.

Is there a recommended length for tests?

Although there is no formal limit regarding test length, it is recommended that you divide your tests into actions and that you use reusable actions in tests, whenever possible. An action should contain no more than a few hundreds steps and, ideally, no more than a few dozen. For more information, see Chapter 15, "Working with Actions."

C

Creating Custom Process Guidance Packages

This chapter guides you through the process of creating custom process guidance packages. You can distribute your custom packages to the QuickTest users in your organization. QuickTest users can then display the processes from your package in QuickTest while they work, to assist them in following your organization's processes and standards.

This chapter includes:

About Process Guidance Packages

A Process Guidance Package is comprised of two entities: the package configuration file and the data files.

- **Package Configuration file**. This XML file defines the **Processes** included in the package and the structure of the **Groups** and **Activities** in each process.
- **Data Files**. A set of HTML files. Each HTML file contains the content for a single activity.

For an overview of process guidance and how it is used in QuickTest, see Chapter 43, "Working with Process Guidance."

Understanding the Package Configuration File

To create a new package, you first create an XML file that describes the processes included in the package and sets the structure of the groups and activities in each process. This structure is displayed as a table of contents for a selected process in the QuickTest **Process Guidance Activities** pane.

Important: Save the configuration file with the name: **Configuration.xml**

The following is an example of a package configuration file that contains two processes:

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessGuidance Name="MyCustomPackage">
   <Process Name="My Process" ID="Process1" DocType="test" Addin="web"
SortLevel="4" >
      <Group Name="New User Overview">
         <Activity Name="Step 1" Address="Step1.html"  />
         <Activity Name="Step 2" Address="Step2.html"  />
      </Group>
   </Process>
   <Process Name="Important Processes" ID="Process2" DocType="test|AA"
SortLevel="3">
      <Group Name="Getting Started">
         <Activity Name="Open" Address="F:\ProcessData\open.html"  />
         <Activity Name="Create" Address="F:\ProcessData\create.html"  />
         <Activity Name="Test" Address="F:\ProcessData\test.html"  />
         <Activity Name="Debug" Address="F:\ProcessData\debug.html"  />
      </Group>
      <Group Name="Finish">
         <Activity Name="Save" Address="F:\ProcessData\save.html"  />
         <Activity Name="Close" Address="F:\ProcessData\close.html"  />
         <Activity Name="Exit" Address="F:\ProcessData\exit.html"  />
      </Group>
   </Process>
</ProcessGuidance>
```

XML Details

The elements and attributes you can use in your package configuration file are described in this section.

- **<Process> Element.** Defines a new process. This element supports the following attributes:
  - **Name**. The name of the process as you want it to appear in the QuickTest Process Guidance pane.
  - **ID**. A unique identification name. This name is used to distinguish between two processes with the same name.
  - **DocType**. Indicates the QuickTest document types for which this process is applicable. If specified, the process is available only when the relevant document type is open.

    In the example above, if a QuickTest user opens a test document, both processes will be available, but if an application area document is opened, only the second process will be available.

    Possible values:
    - **test**. A test document.
    - **AA**. An application area document.
    - **BC**. A business component document.
    - **SBC**. A scripted component document.
  - **Addin**. Indicates the QuickTest add-ins for which this process is applicable. If specified, the process is available only when the relevant add-in is loaded.

    In the example above, the first process will be available only if the Web Add-in is loaded. The second process will always be visible.

    Specify the add-in value using the add-in name as displayed in the Add-in Manager.
  - **SortLevel**. Determines the location of the process within the process list. This list is displayed in the Process Guidance Management dialog box and in the QuickTest **Automation** > **Process Guidance List** menu.

- **<Group> Element**. Defines a new group in the process. This element supports the following attributes:
  - **Name**. Same as the **Name** attribute for the **<Process>** element, as described above.
  - **ID**. Same as the **ID** attribute for the **<Process>** element, as described above.
  - **Addin**. Same as the **Addin** attribute for the **<Process>** element, as described above.
- **<Activity> Element.** Defines an activity within the group.
  - **Name**. Same as the **Name** attribute for the **<Process>** element, as described above.
  - **ID**. Same as the **ID** attribute for the **<Process>** element, as described above.
  - **Addin**. Same as the **Addin** attribute for the **<Process>** element, as described above.
  - **Address**. The path where the relevant HTML data file is located. This can be a local or network path on the file system or an HTTP address. If you specify a relative path, the location is resolved relative to the configuration file location.

Creating Data Files

Each data file contains the HTML content for a single process guidance activity. When an activity link is clicked in the **Process Guidance Activities** pane, the HTML content is displayed in a browser control in the **QuickTest Process Guidance Description** pane.

The package data files can include reference to a **.css** file to display content in your organization's standard style, and can contain any content that can be displayed by a browser.

You can also add special code to your HTML pages to activate QuickTest dialog boxes or jump to other process guidance processes or activities using the QuickTest **UI** automation object. For more information, see the Automation Object Model Reference (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model Reference**).

The HTML files, and any folders or files that the HTML files reference can be stored on the user's local hard drive in a network location on the file system or on a Web server. The package configuration file (the **Address** attribute of each **Activity** element) provides HTML links for each activity.

You should write the HTML file for each activity such that there will be minimum scrolling when the content is displayed in the Process Guidance Description pane at its default size.

If you find that your HTML files are too long, you may want to break them up into multiple process guidance activities to make it easier for your QuickTest users to reference while they work.

Installing Custom Process Guidance Packages in QuickTest

There are two ways to distribute and install custom process guidance packages:

- Install the process guidance package from a zip file
- Install the process guidance package via registry key

Install the process guidance package from a zip file

1. Create a folder that contains the **Configuration.xml** file and all the HTML data files (as well as any files or folders referenced from the HTML files).
2. Zip the folder and then send the **.zip** file to all relevant QuickTest users or store it in a location that they can access.
3. In QuickTest, select **File** > **Process Guidance Management**. The Process Guidance Management dialog box opens.
4. Click the **Add** button and browse to the **.zip** file. The package is added and its processes are displayed in the dialog box.

Install the process guidance package via registry key

1. Prepare the **Configuration.xml** file and the data files.
2. Place the data files in a local or shared network folder or on a Web server. Ensure that the **Address** attribute of the **Activity** elements in the **Configuration.xml** file point to this location.
3. Copy the **Configuration.xml** to a local drive on the QuickTest computer.
4. Open the Registry Editor and find the key:

   HKEY_LOCAL_MACHINE\SOFTWARE\Mercury Interactive\QuickTest Professional\MicTest\ProcessGuidance\ConfFiles

5. Add a value to this key with the path to the **Configuration.xml** file. The next time QuickTest is opened, it will include the new package.

D

Bitmap Checkpoint Customization

Important: This appendix is intended for COM programmers who want to customize the algorithm used to compare bitmaps in bitmap checkpoints.

By default, a bitmap checkpoint compares the actual and expected bitmaps pixel by pixel and fails if there are any differences. QuickTest enables its users to define tolerance levels for bitmap checkpoints to refine the bitmap comparison and make it more flexible. For more information, see "Fine-Tuning the Bitmap Comparison" on page 516.

If you need to further customize the way bitmaps are compared in checkpoints, you can develop custom comparers that compare bitmaps according to your requirements. You develop a custom comparer as a COM object and install and register it on the QuickTest computer. A QuickTest user can then choose to use a custom comparer to perform the comparison in a bitmap checkpoint (on a per checkpoint basis).

This chapter includes:

About Bitmap Checkpoint Customization

You implement bitmap checkpoint customization by developing custom comparers. A custom comparer is a COM object that you develop to run the bitmap comparison in a bitmap checkpoint according to a specific algorithm. The COM object that you develop must implement interfaces that QuickTest provides in a type library, and register to the component category that QuickTest defines for bitmap comparers. The type library (**BitmapComparer.tlb**) and the category ID (defined in **ComponentCategory.h**) are available in **<QuickTest installation folder>\dat\BitmapCPCustomization**.

When a QuickTest user creates or edits a bitmap checkpoint, QuickTest displays any registered custom comparers in the Bitmap Checkpoint Properties dialog box (in addition to the QuickTest default comparer). The user can then select a comparer according to the testing requirements of the specific application or bitmap being tested. For more information about using custom comparers in QuickTest, see "Custom Comparer Options in the Bitmap Checkpoint Properties Dialog Box" on page 527.

Before you begin to develop a custom comparer, carefully consider the information in "Considerations for Developing Custom Comparers" below.

You can find an example of a situation where bitmap checkpoint customization enhanced the use of bitmap checkpoints, in "Use-Case Scenario: Handling Images Whose Location in the Application Changes" on page 1577.

Considerations for Developing Custom Comparers

- To develop a custom comparer you must understand image processing and know how to develop COM objects.
- You can implement a custom comparer using any language and development environment that supports creating COM objects.
- Custom comparers run within the QuickTest context. You must therefore exercise care when developing your custom comparer, as its behavior and performance will affect the behavior and performance of QuickTest.
- The custom comparer must be installed and registered on any computer that runs a test with a bitmap checkpoint using the custom comparer.

- Before installing and registering a new version of a custom comparer, unregister the existing comparer.
- More than one custom comparer can be installed and registered on the same QuickTest computer. In the Bitmap Checkpoint Properties dialog box, QuickTest displays all of the available custom comparers, and the QuickTest default comparer. The QuickTest user can select the appropriate comparer to use for each bitmap checkpoint.
- The computer that runs the custom comparer must have installed the runtime environment associated with the configuration in which the custom comparer DLL was built.
- You create the custom comparer DLL using a specific development environment version; the computer on which this DLL runs must have the corresponding runtime environment installed.

Use-Case Scenario: Handling Images Whose Location in the Application Changes

Ben is a quality assurance engineer who is experienced in using QuickTest, and often uses bitmap checkpoints to test the appearance of different icons or pictures in the user interface he is testing. He does not have a programming background.

Joanne is a software engineer who is experienced in image processing and is familiar with COM programming.

When Ben began testing the user interface of a furniture purchasing application, he created a bitmap checkpoint to test that the pictures of the items on sale were displayed properly. In the checkpoint, he captured an image of the pane in the application that contained the pictures he wanted to test. Ben found that the bitmap checkpoint often failed, even though the graphic images displayed in the application during the run seemed identical to the ones he had captured when creating the checkpoint.

Ben reviewed the actual, expected, and difference bitmaps displayed in the test results. He also took a closer look at the application's user interface. The application contained three panes. The left pane displayed general information, the middle pane displayed the pictures of the items on sale, and the right pane displayed the corresponding list of items and relevant details. Ben found that depending on the information displayed in the left pane, the images in the middle pane sometimes shifted slightly one way or the other within the pane. While the images themselves were still identical, their changed location was causing the bitmap checkpoint to fail.

Ben did not want to use pixel tolerance to address this issue because he wanted the checkpoint to fail when the pixels within the images themselves were not identical.

When Ben mentioned his problem to a co-worker, she suggested that bitmap checkpoint customization could solve the problem, and referred him to Joanne. Joanne developed a custom comparer that would accept as input the number of pixels that the images should be allowed to shift without failing the checkpoint. The bitmap comparison that Joanne designed would pass the checkpoint only if the images were identical and they had all shifted by the same number of pixels. This way, Ben knew that his checkpoint would still catch incorrect images and cases where the application's interface looked bad because the images were no longer aligned.

Ben installed and registered the custom comparer on his QuickTest computer, and then selected the new custom comparer for his bitmap checkpoint. After some experimenting, he found the optimal number of pixels to enter in the configuration string, so that significant changes in the application's interface were detected, but insignificant shifting of the images did not cause the checkpoint to fail.

After Ben successfully used this custom comparer for a while, his company decided to install and register it on all of the QuickTest computers. The custom comparer would now be available to everyone in the quality assurance team to use for similar situations.

Developing a Custom Bitmap Comparer

To develop a custom comparer, you create a COM object that implements the QuickTest bitmap checkpoint comparer interfaces (described on page 1585) to perform the following tasks:

- Accept input from QuickTest and perform the bitmap comparison.
- Provide comparison results to QuickTest.
- (Optionally) Provide information that QuickTest can display in the Bitmap Checkpoint Properties dialog box when a user creates or edits a bitmap checkpoint.

For more information, see "How to Develop a Custom Comparer" on page 1580.

For QuickTest to recognize the custom comparer, it must be registered to the component category that QuickTest defines for bitmap comparers. Depending on how you implement your custom comparer, you can design the comparer to register itself when it is installed, or you can provide an additional program that needs to be run at the time of installation. For more information, see "Installing Your Custom Comparer and Registering it to QuickTest" on page 1582.

Perform the tutorial in "Tutorial: Creating a Custom Comparer" on page 1589 to learn how to create and use a custom comparer. You can then create your own custom comparers in much the same way.

QuickTest provides sample custom comparers that you can use as a reference or template when developing custom comparers. For more information, see "Using the Bitmap Checkpoint Customization Samples" on page 1600.

How to Develop a Custom Comparer

In the COM object that you develop, reference the type library that QuickTest provides (located in **<QuickTest installation folder>\dat\BitmapCPCustomization\BitmapComparer.tlb**) and implement the interfaces to perform the tasks described in the following sections:

- "Accepting Input and Comparing the Bitmaps" on page 1580
- "Providing Comparison Results to QuickTest" on page 1581
- "Providing Information for the Bitmap Checkpoint Properties Dialog Box" on page 1581

Accepting Input and Comparing the Bitmaps

QuickTest calls the **CompareBitmaps** method in the **IVerifyBitmap** interface (described on page 1585) to pass the expected and actual bitmaps to the custom comparer for comparison. Implement the **CompareBitmaps** method to perform the following:

- Accept and compare two bitmaps according to a predefined algorithm that you define based on the testing requirements.
- Accept a text string that can contain configuration information provided by the QuickTest user (in the Bitmap Checkpoint Properties dialog box), and use it in the comparison. For example, the string could contain tolerance specifications, acceptable deviations in size or location of the image, or any other information that you want to affect the comparison.

  The string can have any format you choose (XML, comma separated, INI file style, and so on). Make sure that the documentation you provide for the custom comparer describes the format. The configuration input that the QuickTest user enters in the Bitmap Checkpoint Properties dialog box must conform to this format.

Providing Comparison Results to QuickTest

QuickTest displays the results of bitmap checkpoints in the Test Results window. When you implement the **IVerifyBitmap** interface (described on page 1585) to compare the bitmaps, you must also return the following information:

- Whether the bitmaps match and the checkpoint should pass.
- A text string that QuickTest displays in the test results.

  The purpose of this string is to provide information about the comparison to the QuickTest user, but while you develop and test your comparer, you can use this string for debugging purposes as well.
- A bitmap that visually represents the difference between the actual and expected bitmaps.

  The purpose of this bitmap is to help the QuickTest user understand why the checkpoint failed. The custom comparer can create this bitmap using any visualization approach you choose. For example, the default QuickTest comparer creates a black-and-white bitmap containing a black pixel for every pixel that is different in the two images.

Providing Information for the Bitmap Checkpoint Properties Dialog Box

When a QuickTest user selects a custom comparer in the Bitmap Checkpoint Properties dialog box, QuickTest displays a **Configuration options** text box, and, optionally, a link to documentation provided for the custom comparer. For more information, see "Custom Comparer Options in the Bitmap Checkpoint Properties Dialog Box" on page 527. To support these options, you can implement the **IBitmapCompareConfiguration** interface (described on page 1587) to provide the following:

- A default configuration string that QuickTest displays in the **Configuration options** box in the Bitmap Checkpoint Properties dialog box.

  The format of this string must be the same as the format of the configuration string that the comparer expects as input.

➤ Documentation about the comparer that the QuickTest user can access from the Bitmap Checkpoint Properties dialog box.

The documentation can be in any format that you choose. QuickTest opens the documentation using the program associated with the provided file type on the user's computer. Therefore, you should provide the documentation in a format for which you expect the QuickTest user to have the necessary program.

The documentation should provide the QuickTest user with the following information:

- ➤ The type of comparison the custom comparer performs (to enable the user to determine when to use it to run a bitmap checkpoint).
- ➤ The required format for the configuration string and the possible values it can contain.
- ➤ An explanation of the comparison result information that is displayed in the test results (text string and difference bitmap).

Installing Your Custom Comparer and Registering it to QuickTest

The custom comparer that you develop needs to be installed on any computer that runs a test that includes a bitmap checkpoint that uses the custom comparer.

Make sure that when the custom comparer is installed, the documentation that you provide for the QuickTest user is placed in the location that you specified in the **GetHelpFilename** method. (For more information see, "The GetHelpFilename Method" on page 1588.)

In addition, for QuickTest to recognize the COM object that you create as a custom comparer, you must register it to the component category for QuickTest bitmap comparers.

You register a COM object to a component category by listing the relevant component category ID as a registry key under the COM object's **HKEY_CLASSES_ROOT\CLSID\<Object's CLSID>\Implemented Categories** key.

The component category ID must be registered under the **HKEY_CLASSES_ROOT\Component Categories** key. When QuickTest is installed, it adds the component category ID for QuickTest bitmap comparers as a registry key in this location.

The component category ID for QuickTest bitmap comparers, **CATID_QTPBitmapComparers**, is defined in **<QuickTest installation folder>\dat\BitmapCPCustomization\ComponentCategory.h**.

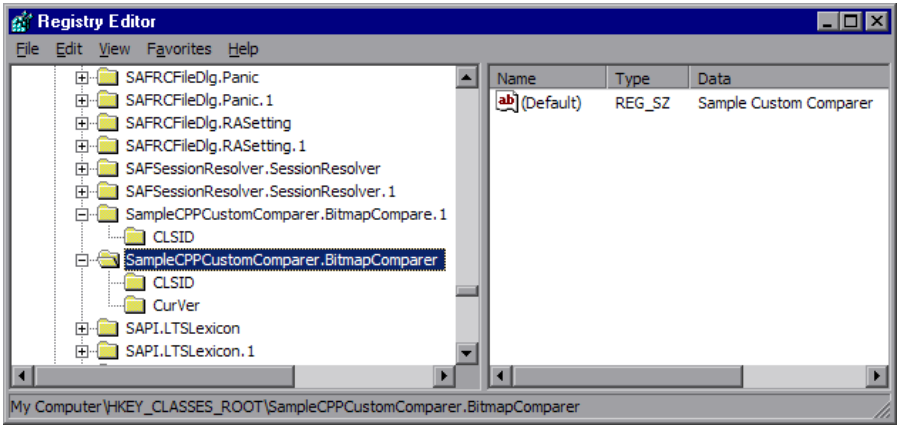
When you design your custom comparer, you must ensure that when it is installed on the QuickTest computer, it is also registered to the component category for QuickTest bitmap comparers. This can be achieved in different ways.

For example:

- If you develop your custom comparer in C++ using Microsoft Visual Studio, you can modify the **DllRegisterServer** and **DllUnregisterServer** methods to handle this registration. These methods are called when you run a DLL using the **regsvr32.exe** program. You can see an example of this type of implementation in step 6 of "Tutorial: Creating a Custom Comparer", on page 1597.
- If you develop your custom comparer in an environment that does not enable you to modify the registration methods, you can add an additional program that handles this registration and instruct users who install the custom comparer to run this program as well. You can see an example of this type of implementation in the Visual Basic sample custom comparer that QuickTest provides. For more information, see "Sample Custom Comparer Registration" on page 1600.

Setting the Custom Comparer Name

QuickTest displays the name of the custom comparer in the Bitmap Checkpoint Properties dialog box and in the Test Results window. The name that QuickTest uses is the value (in the registry) of the default property of the custom comparer ProgID key under the HKEY_CLASSES_ROOT key. For example, in the image below, the name of the custom comparer is **Sample Custom Comparer**.



- If you develop your custom comparer in C++ using Microsoft Visual Studio, you can specify this name in the **Type** box in the ATL Simple Object Wizard.
- If you develop the custom comparer in Visual Basic, this value is automatically set to the COM object's ProgID. If you want to modify the custom comparer name, you can edit it manually in the registry after the comparer is installed, or design the program that performs the installation and registration to edit this value as well.

The Bitmap Checkpoint Comparer Interfaces

Your custom comparer must implement the interfaces described in this section. QuickTest calls these interfaces' methods when creating or running a bitmap checkpoint that uses your custom comparer.

The IVerifyBitmap Interface

Implement the **CompareBitmaps** method to perform the bitmap comparison for the checkpoint.

The CompareBitmaps Method

The **CompareBitmaps** method receives the actual and expected bitmaps that need to be compared for the bitmap checkpoint, and a string that can contain configuration input for the custom comparer.

The method must compare the bitmaps according to the comparison algorithm for which this custom comparer is designed, and return the results to QuickTest.

The results include:

- An indication whether the bitmaps match and the checkpoint should pass.
- A text string that contains information about the results of the bitmap comparison.
- A bitmap that reflects the differences between the actual and expected bitmaps.

QuickTest displays the results that this method returns in the Test Results window. For more information, see "Analyzing Bitmap Checkpoint Results" on page 1033.

Method syntax:

```
HRESULT CompareBitmaps ([in] IPictureDisp* pExpected,
                        [in] IPictureDisp* pActual,
                        [in] BSTR bstrConfiguration,
                        [out] BSTR* pbstrLog,
                        [out] IPictureDisp** ppDiff,
                        [out, retval] VARIANT_BOOL* pbMatch);
```

Method Parameters:

- *pExpected*. A picture object (input).

  The expected bitmap stored in the checkpoint.

- *pActual*. A picture object (input).

  The actual bitmap captured from the application being tested.

- *bstrConfiguration*. A text string (input).

  A string that contains configuration input for the custom comparer. This is the string displayed in the **Configuration options** box in the Bitmap Checkpoint Properties dialog box.

  The string can be the default configuration string that the custom comparer provides to QuickTest in the **GetDefaultConfigurationString** method described below, or an input string entered by the QuickTest user.

  The **bstrConfiguration** string can have any format you choose (XML, comma separated, ini file style, and so on). Make sure that the default configuration string returned by the **GetDefaultConfigurationString** method matches the format expected in the **CompareBitmaps** method. Additionally, make sure that the documentation you provide for your custom comparer explains the format that the QuickTest user must use when editing this string in the **Configuration options** box.

- *pbstrLog*. A text string (output).

  A string that contains information about the results of the bitmap comparison. QuickTest displays this string in the Test Results window.

- *ppDiff*. A picture object (output).

  A bitmap (created by the custom comparer) that reflects the difference between the actual and expected bitmaps. QuickTest displays this bitmap in the Test Results window along with the actual and expected bitmaps.

➤ *pbMatch*. A boolean value (output).

A value that indicates whether the bitmaps match and the checkpoint should pass.

Possible values:

VARIANT_TRUE. Actual and expected bitmaps match, checkpoint passes.

VARIANT_FALSE. Actual and expected bitmaps do not match, checkpoint fails.

Return Value

The HRESULT that this method returns indicates whether the comparison ran successfully (and not whether the bitmaps match).

The IBitmapCompareConfiguration Interface

Implement the methods in this interface to support the custom comparer options that QuickTest displays in the Bitmap Checkpoint Properties dialog box. For more information, see "The Bitmap Checkpoint Properties Dialog Box" on page 522.

The GetDefaultConfigurationString Method

The **GetDefaultConfigurationString** method must return the default configuration string for your custom comparer. For more information on configuration strings, see "Accepting Input and Comparing the Bitmaps" on page 1580.

QuickTest displays this string in the **Configuration options** box in the Bitmap Checkpoint Properties dialog box when a user creating a new bitmap checkpoint selects your custom comparer.

If the QuickTest user does not modify the configuration string in the dialog box, the string provided by **GetDefaultConfigurationString** is passed to the custom comparer's **CompareBitmaps** method. You must therefore make sure that the default configuration string matches the format that your custom comparer expects to receive in the **CompareBitmaps** method.

Method syntax:

```
HRESULT GetDefaultConfigurationString ([out, retval] BSTR* pbstrConfiguration);
```

The GetHelpFilename Method

The **GetHelpFilename** method must return a path to the documentation that contains information about your custom comparer for QuickTest users.

QuickTest displays the documentation when a user selects your custom comparer in the Bitmap Checkpoint Properties dialog box and clicks **Details**. Make sure that when your custom comparer is installed, the documentation that you provide is installed in the location specified by the **GetHelpFilename** method.

The path can be one of the following:

- A full path to a file.
- A relative path to a file (QuickTest searches for this path relative to **<QuickTest installation folder>\bin**).
- A URL.

If you do not provide documentation for your custom comparer, this method should return the HRESULT E_NOTIMPL. For more information on the type of information you should provide, see "Providing Information for the Bitmap Checkpoint Properties Dialog Box" on page 1581.

Method syntax:

```
HRESULT GetHelpFilename ([out, retval] BSTR* pbstrFilename);
```

Tutorial: Creating a Custom Comparer

This tutorial walks you step-by-step through the process of creating a custom comparer in C++ using Microsoft Visual Studio. The custom comparer you create is similar to the sample custom comparer provided with QuickTest. You can create your own custom comparers in a similar way. For more information about the sample custom comparer, see "Using the Bitmap Checkpoint Customization Samples" on page 1600.

By following the instructions in this section, you create a COM object that:

- Implements the **CompareBitmaps** method to receive two bitmaps to compare and a configuration string, compare the (size of) the two bitmaps, and return the necessary results.
- Implements the **GetDefaultConfigurationString** method and the **GetHelpFilename** method, to return the information that QuickTest displays in the Bitmap Checkpoint Properties dialog box.
- Registers to the component category for QuickTest bitmap comparers.

When the design of your custom comparer is complete, you can install and register it and use it in QuickTest to run a bitmap checkpoint.

Note: Depending on the version of Microsoft Visual Studio that you use to perform the tutorial, the command names may be different.

To practice creating a custom comparer for bitmap checkpoints:

1 Create a new ATL project—SampleCPPCustomComparer.

a In Microsoft Visual Studio, select **New** > **Project**. The New Project dialog box opens.

b Select the **ATL Project** template, enter SampleCPPCustomComparer in the **Name** box for the project, and click **OK**. The New ATL Project wizard opens.

c In **Application Settings**, make sure that the **Attributed** option is not selected, and click **Finish**.

2 Create a new class—CBitmapComparer.

a In the class view, select the **SampleCPPCustomComparer** project, right-click, and select **Add** > **Class**. The Add Class dialog box opens.

b Select **ATL Simple Object** and click **Add**. The ATL Simple Object Wizard opens.

c In the **Short name** box, enter BitmapComparer. The wizard uses this name to create the names of the class, the interface, and the files that it creates.

d In the **Type** box, enter Sample Custom Comparer. This is the custom comparer name that QuickTest will display in the Bitmap Checkpoint Properties dialog box and in the test results. For more information, see "Setting the Custom Comparer Name" on page 1584.

e Click **Finish**. The wizard creates the necessary files for the class that you added, including **.cpp** and **.h** files with implementation of **CBitmapComparer** class.

3 Define that the CBitmapComparer class implements the bitmap checkpoint comparer interfaces.

a In the class view, select **CBitmapComparer**, right-click, and select **Add** > **Implement Interface**. The Implement Interface wizard opens.

b In the **Implement interface from** option, select **File**. Browse to or enter the location of the QuickTest bitmap checkpoint comparer type library. The type library is located in: **<QuickTest installation folder>\dat\BitmapCPCustomization\BitmapComparer.tlb**.

The wizard displays the interfaces available in the selected type library, **IBitmapCompareConfiguration** and **IVerifyBitmap**.

c Add both interfaces to the list of interfaces to implement, and click **Finish**.

In the **BitmapComparer.h** file, the wizard adds the declarations, classes, and method stubs that are necessary to implement the interfaces. In subsequent steps you will need to add implementation to these method stubs.

Note: In Microsoft Visual Studio 2005, the wizard generates the signature for the **CompareBitmaps** method in the **IVerifyBitmap** interface incorrectly. To enable your project to compile correctly, manually change the type of the last argument (*pbMatch*) from BOOL* to VARIANT_BOOL*.

---

4 Move the function bodies for the bitmap checkpoint comparer interface methods from BitmapComparer.h to BitmapComparer.cpp.

a Open the **BitmapComparer.h** and **BitmapComparer.cpp** files.

b In **BitmapComparer.h**, create declarations for the bitmap checkpoint comparer interface methods (based on the function bodies that the wizard created): **CompareBitmaps**, **GetDefaultConfigurationString**, and **GetHelpFilename**.

c Move the function bodies that the wizard created for the bitmap checkpoint comparer interface methods from the **BitmapComparer.h** file to the **BitmapComparer.cpp** file.

At the end of this step, **BitmapComparer.cpp** and **BitmapComparer.h** should contain the following code:

```
// BitmapComparer.cpp : Implementation of CBitmapComparer
#include "stdafx.h"
#include "BitmapComparer.h"

// CBitmapComparer
// IBitmapCompareConfiguration Methods
STDMETHODIMP CBitmapComparer::GetDefaultConfigurationString
                                            (BSTR * pbstrConfiguration)
{
        return E_NOTIMPL;
}
STDMETHODIMP CBitmapComparer::GetHelpFilename(BSTR * pbstrFilename)
{
        return E_NOTIMPL;
}

// IVerifyBitmap Methods
STDMETHODIMP CBitmapComparer::CompareBitmaps
                        (IPictureDisp * pExpected, IPictureDisp * pActual,
                        BSTR bstrConfiguration, BSTR * pbstrLog,
                        IPictureDisp * * ppDiff, VARIANT_BOOL * pbMatch)
{
        return E_NOTIMPL;
}
```

```
// BitmapComparer.h : Declaration of the CBitmapComparer
#pragma once
#include "resource.h"       // main symbols
#include "SampleCPPCustomComparer.h"
// CBitmapComparer
class ATL_NO_VTABLE CBitmapComparer :
	public CComObjectRootEx<CComSingleThreadModel>,
	public CComCoClass<CBitmapComparer, &CLSID_BitmapComparer>,
	public IDispatchImpl<IBitmapComparer, &IID_IBitmapComparer,
			&LIBID_SampleCustomComparerLib, /*wMajor =*/ 1, /*wMinor =*/ 0>,
	public IDispatchImpl<IBitmapCompareConfiguration,
			&__uuidof(IBitmapCompareConfiguration),
			&LIBID_BitmapComparerLib, /* wMajor = */ 1, /*wMinor =*/ 0>,
	public IDispatchImpl<IVerifyBitmap, &__uuidof(IVerifyBitmap),
			&LIBID_BitmapComparerLib, /* wMajor = */ 1, /*wMinor =*/ 0>
{
 public:
	CBitmapComparer()
	{
	}
	DECLARE_REGISTRY_RESOURCEID(IDR_BITMAPCOMPARER)
	BEGIN_COM_MAP(CBitmapComparer)
		COM_INTERFACE_ENTRY(IBitmapComparer)
		COM_INTERFACE_ENTRY2(IDispatch, IBitmapCompareConfiguration)
		COM_INTERFACE_ENTRY(IBitmapCompareConfiguration)
		COM_INTERFACE_ENTRY(IVerifyBitmap)
	END_COM_MAP()
	DECLARE_PROTECT_FINAL_CONSTRUCT()
	HRESULT FinalConstruct()
	{
		return S_OK;
	}
	void FinalRelease()
	{}
	// IBitmapCompareConfiguration Methods
public:
	STDMETHOD(GetDefaultConfigurationString)(BSTR * pbstrConfiguration);
	STDMETHOD(GetHelpFilename)(BSTR * pbstrFilename);
	// IVerifyBitmap Methods
public:
	STDMETHOD(CompareBitmaps)(IPictureDisp * pExpected,
			IPictureDisp * pActual, BSTR bstrConfiguration, BSTR * pbstrLog,
			IPictureDisp * * ppDiff, VARIANT_BOOL * pbMatch);
};
OBJECT_ENTRY_AUTO(__uuidof(BitmapComparer), CBitmapComparer)
```

5 Implement the bitmap checkpoint comparer interface methods to customize the bitmap checkpoint as required.

In this tutorial, you implement a custom comparer similar to the sample custom comparer provided with QuickTest. For more information about the sample custom comparer, see "Using the Bitmap Checkpoint Customization Samples" on page 1600.

When you create your own custom comparers, this is the step during which you design the custom comparer logic. You define the configuration input that it can receive, the algorithm that it uses to compare the bitmaps, and the output that it provides.

In the **BitmapComparer.cpp** file, add #include <atlstr.h>, and implement the bitmap checkpoint comparer interface methods as follows:

➤ The **GetDefaultConfigurationString** method:

```
STDMETHODIMP CBitmapComparer::GetDefaultConfigurationString
                                              (BSTR * pbstrConfiguration)
{
      CComBSTR bsConfig("MaxSurfAreaDiff=140000");
      *pbstrConfiguration = bsConfig.Detach();
      return S_OK;
}
```

➤ The **GetHelpFilename** method: (If you copy and paste the code from this PDF, make sure to remove the line break and tabs from the filename string.)

```
STDMETHODIMP CBitmapComparer::GetHelpFilename(BSTR * pbstrFilename)
{
      CComBSTR bsFilename ("..\\samples\\BitmapCPSample\\CPPCustomComparer\\
            SampleComparerDetails.txt");
      *pbstrFilename = bsFilename.Detach();
      return S_OK;
}
```

Note: When the **GetHelpFilename** method returns a relative path, QuickTest searches for this path relative to **<QuickTest installation folder>\bin**. The implementation above instructs QuickTest to use the documentation file provided with the CPP sample custom comparer.

➤ The **CompareBitmaps** method:

```
STDMETHODIMP CBitmapComparer::CompareBitmaps
                                    (IPictureDisp * pExpected, IPictureDisp * pActual,
                                    BSTR bstrConfiguration, BSTR * pbstrLog,
                                    IPictureDisp * * ppDiff, VARIANT_BOOL * pbMatch)
{
      HRESULT hr = S_OK;
      if (!pExpected || !pActual)
            return S_FALSE;
      CComQIPtr<IPicture> picExp(pExpected);
      CComQIPtr<IPicture> picAct(pActual);

      // Try to get HBITMAP from IPicture
      HBITMAP HbmpExp, HbmpAct;
      hr = picExp->get_Handle((OLE_HANDLE*)&HbmpExp);
      if (FAILED(hr))
            return hr;
      hr = picAct->get_Handle((OLE_HANDLE*)&HbmpAct);
      if (FAILED(hr))
            return hr;
      BITMAP ExpBmp = {0};
      if( !GetObject(HbmpExp, sizeof(ExpBmp), &ExpBmp) )
            return E_FAIL;
      BITMAP ActBmp = {0};
      if( !GetObject(HbmpAct, sizeof(ActBmp), &ActBmp) )
            return E_FAIL;

      CString s, tol;
      tol = bstrConfiguration;
      int EPos = tol.ReverseFind('=');
      tol = tol.Right(tol.GetLength() - EPos - 1);
      int maxSurfaceAreaDiff = _ttoi(tol);
      // Set output parameters
      CComPtr<IPictureDisp> Diff(pActual);
      *ppDiff = Diff;
      int DiffPixelsNumber = abs (ExpBmp.bmHeight * ExpBmp.bmWidth -
                                    ActBmp.bmHeight * ActBmp.bmWidth);
      *pbMatch = DiffPixelsNumber <= maxSurfaceAreaDiff;
      s.Format(_T("The number of different pixels is: %d."), DiffPixelsNumber);
      CComBSTR bs (s);
      *pbstrLog = bs.Detach();
      return hr;
}
```

6 Design your custom comparer to register to the component category for QuickTest bitmap comparers.

For QuickTest to recognize the COM object that you create as a custom comparer, you must register it to the component category for QuickTest bitmap comparers. The component category ID is defined in **<QuickTest installation folder>\dat\BitmapCPCustomization\ComponentCategory.h**.

You can implement this registration in the **DllRegisterServer** and **DllUnregisterServer** methods in the **SampleCPPCustomComparer.cpp** file that the wizard created as part of your project. These methods are called when you run a DLL using the **regsvr32.exe** program.

a Add the **<QuickTest installation folder>\dat\BitmapCPCustomization** folder to your project's include path.

b Open the **SampleCPPCustomComparer.cpp** file and add the following line: #include "ComponentCategory.h"

c In the **SampleCPPCustomComparer.cpp** file, modify the **DllRegisterServer** and **DllUnregisterServer** methods created by the wizard, to contain the following code:

```
STDAPI DllRegisterServer(void)
{
      // registers object, typelib and all interfaces in typelib
      HRESULT hr = _AtlModule.DllRegisterServer();

     CComPtr<ICatRegister> spReg;
     hr = spReg.CoCreateInstance
               (CLSID_StdComponentCategoriesMgr, 0, CLSCTX_INPROC);
     if (FAILED(hr))
          return hr;

     // register comparer to the QuickTest bitmap comparers category
     CATID catid = CATID_QTPBitmapComparers;
     hr = spReg->RegisterClassImplCategories(CLSID_BitmapComparer, 1, &catid);

     return hr;
}
```

```
STDAPI DllUnregisterServer(void)
{
	HRESULT hr = _AtlModule.DllUnregisterServer();
	CComPtr<ICatRegister> spReg;
	hr = spReg.CoCreateInstance
		(CLSID_StdComponentCategoriesMgr, 0, CLSCTX_INPROC);
	if (FAILED(hr))
		return hr;

	// unregister comparer from the QuickTest bitmap comparers category
	CATID catid = CATID_QTPBitmapComparers;
	hr = spReg->UnRegisterClassImplCategories(CLSID_BitmapComparer, 1, &catid);

	return hr;
}
```

Note the second section in these methods, that handles registration to the component category for QuickTest bitmap comparers—**CATID_QTPBitmapComparers**.

7 Compile your DLL and run it using the regsvr32.exe program.

Your custom comparer can now be used in QuickTest for bitmap checkpoints.

8 Use your custom comparer for bitmap checkpoints in QuickTest.

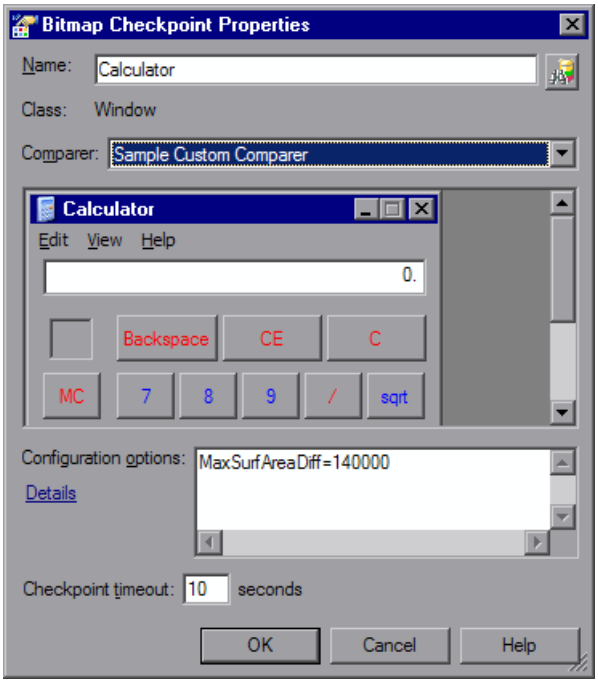
For more information on how to work with bitmap checkpoints, see Chapter 19, "Checking Bitmaps."

a Open QuickTest and create a bitmap checkpoint on the Windows Calculator application (Standard view).

The Bitmap Checkpoint Properties dialog box includes the **Comparer** option, in which you can select the QuickTest default comparer or your sample custom comparer.

b Change the Calculator view to **Scientific**. The size of the calculator object is now larger. Run the checkpoint using the default QuickTest comparer. The checkpoint fails.

c Edit the checkpoint and select **Sample Custom Comparer** in the **Comparer** box.



In the Bitmap Checkpoint Properties dialog box, in the **Configuration options** box, you can see the default configuration string returned by the **GetHelpFilename** method: MaxSurfAreaDiff=140000

If you click **Details**, the text file containing documentation for the sample custom comparer opens.

The comparer you designed in this exercise checks how different the expected and actual bitmaps are in size, and fails the checkpoint if the difference is greater than the number of pixels defined in the configuration string. If you run the checkpoint using default MaxSurfAreaDiff value, the checkpoint passes, because the difference in the total size of the calculator object when it is set to different views is less than 140000 pixels (the difference is approximately 80000 pixels). If you set MaxSurfAreaDiff to 70000, the checkpoint fails.

View the test results to see the text string and difference bitmap that your custom comparer provides to QuickTest after the comparison.

Using the Bitmap Checkpoint Customization Samples

QuickTest provides source files that implement a sample custom comparer in different languages. You can study these examples to help you learn about QuickTest bitmap checkpoint customization, or use them as a reference or template when you develop your own custom comparers. The source files are provided in C++ and in Visual Basic. Both projects generate a similar custom comparer.

The samples are located under **<QuickTest installation folder>\samples\BitmapCPSample**. To open the C++ project, use Microsoft Visual Studio 2003 or later. To open the Visual Basic project, use Microsoft Visual Studio 6.0. You can compile the custom comparer and install it on the QuickTest computer to see how this custom comparer works.

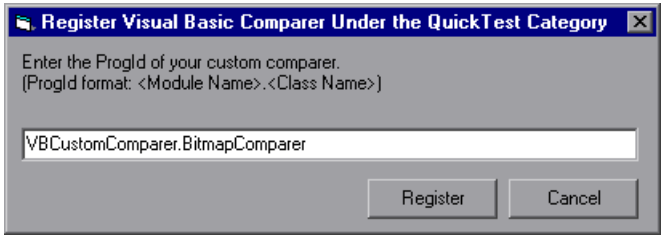
Sample Custom Comparer Registration

After you build the DLL for the custom comparer, run it with the **regsvr32.exe** program to install it on the computer.

The C++ sample sources implement registering the custom comparer to QuickTest in the **DllRegisterServer** and **DllUnregisterServer** methods. Therefore, if you used the C++ project to create the DLL, running the DLL will also register the custom comparer to the component category for QuickTest bitmap comparers.

Registering the custom comparer to the component category for QuickTest bitmap comparers is not implemented in the Visual Basic sample project. Therefore, the Visual Basic sample also includes an additional tool that you must run after installing the custom comparer, to register the custom comparer to the component category for QuickTest bitmap comparers. For more information, see “Installing Your Custom Comparer and Registering it to QuickTest” on page 1582.

You can run the Visual Basic Comparer Registration Tool from **<QuickTest installation folder>\samples\BitmapCPSample\VBCustomComparer\RegisterCategory.exe** (if you copy and paste this path from the PDF, make sure to remove the line break).



In the dialog box that opens, enter the ProgID of the custom comparer and click **Register**.

Sample Custom Comparer Name

The name under which the custom comparer appears in QuickTest is different, depending on whether you generate if from the C++ project or the Visual Basic project:

- If generated from the C++ project, the name displayed for the comparer in QuickTest is **Custom QTP Bitmap Comparer**.
- If generated from the Visual Basic project, the comparer name is its ProgId—**VBCustomComparer.BitmapComparer**.

For more information, see “Setting the Custom Comparer Name” on page 1584.

Sample Custom Comparer Functionality

After you install and register the sample custom comparer, you can select it in the Bitmap Checkpoint Properties dialog box in QuickTest, and use it to perform bitmap checkpoints.

- The default configuration string that the sample comparer returns (and QuickTest displays in the Bitmap Checkpoint Properties dialog box) is MaxSurfAreaDiff=140000.
- The sample custom comparer does not compare the content of the actual and expected bitmaps. It compares the total number of pixels they contain. For configuration input, this comparer expects a string that defines the MaxSurfAreaDiff parameter. The comparer fails the checkpoint if the difference in total number of pixels is greater than the number defined for MaxSurfAreaDiff.

Tip: You can run bitmap checkpoints on the Windows Calculator application to test the behavior of the sample comparer. Set the Calculator view alternately to **Standard** or **Scientific**, to obtain different size bitmaps for the same object.

- The documentation provided with this sample comparer (and opened from the Bitmap Checkpoint Properties dialog box) is the **SampleComparerDetails.txt** text file located in **<QuickTest installation folder>\samples\BitmapCPSample\CPPCustomComparer**.
- For the test results, this sample bitmap custom comparer returns the actual bitmap as the difference bitmap. In addition, it provides a text string that specifies the difference in total number of pixels. QuickTest displays this string in the test results.

Index

This index contains entries from both volumes of the *HP QuickTest Professional User Guide*.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

Q

S

T

U

V

W

X

Z